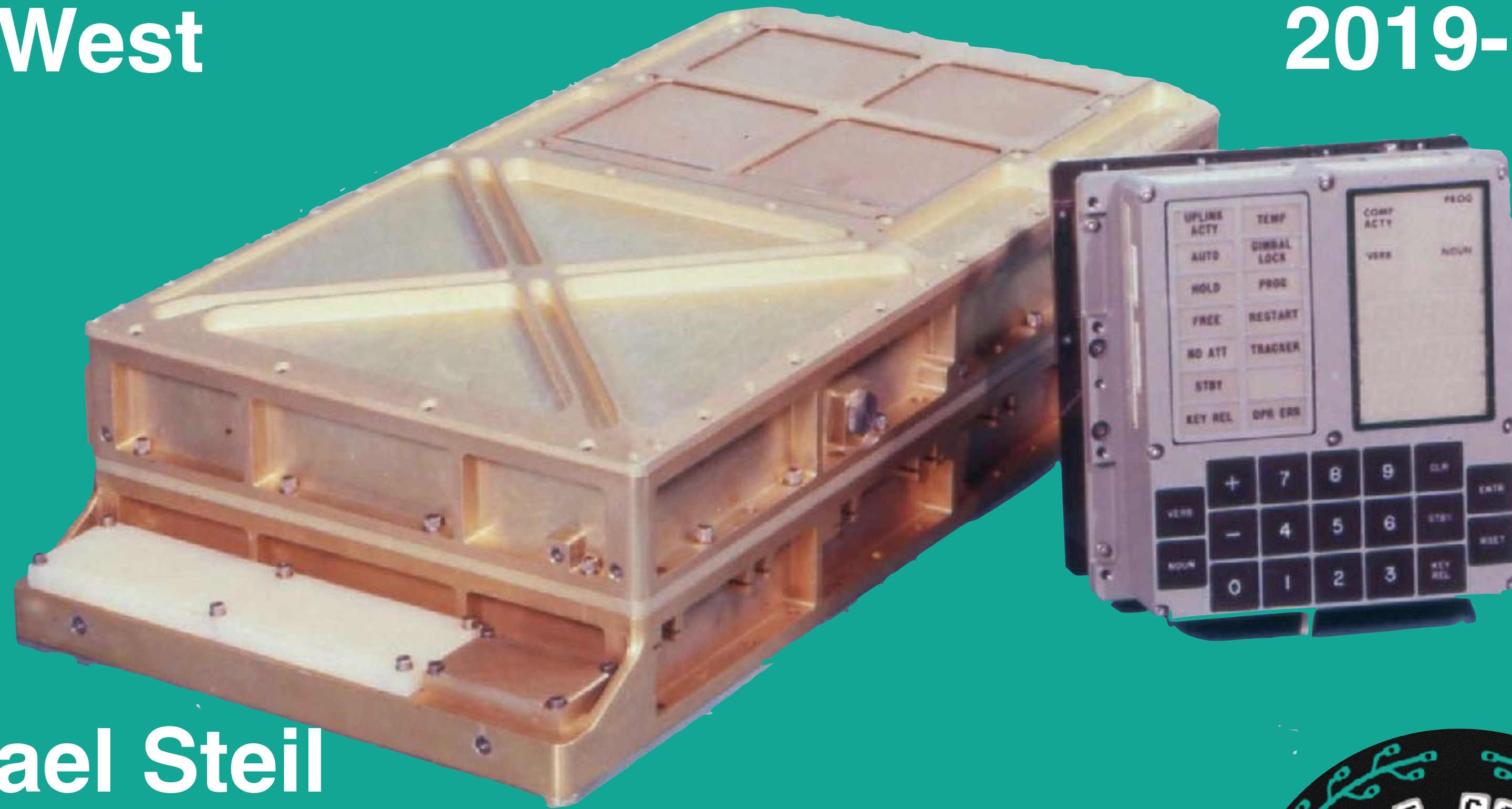


VCF West

2019-08-04



Michael Steil

# The Apollo Guidance Computer



@pagetable

<http://www.pagetable.com/>



# Apollo Guidance Computer





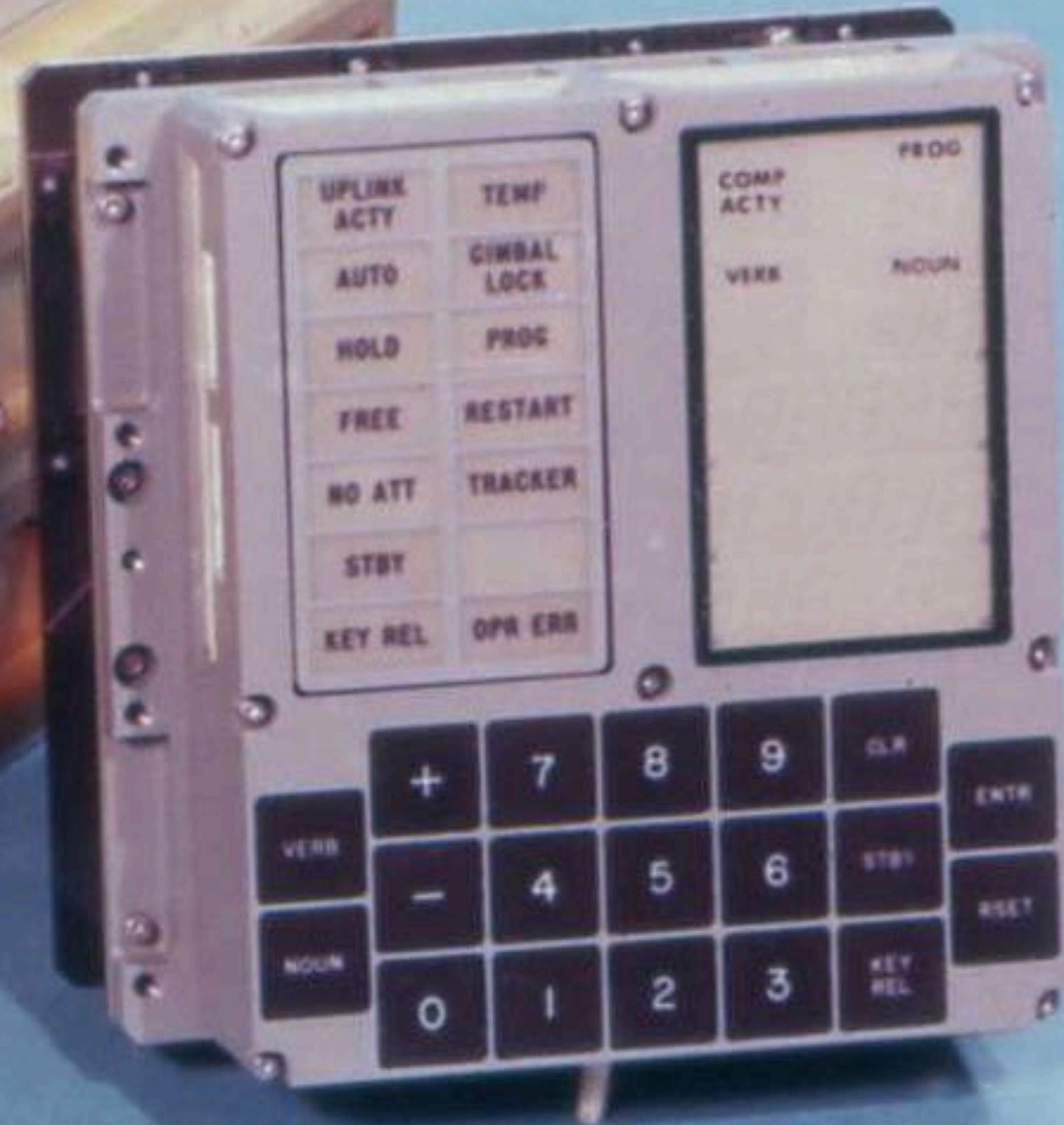
# Apollo Guidance Computer





# Apollo Guidance Computer

MIT  
1966  
\$200,000  
57 built

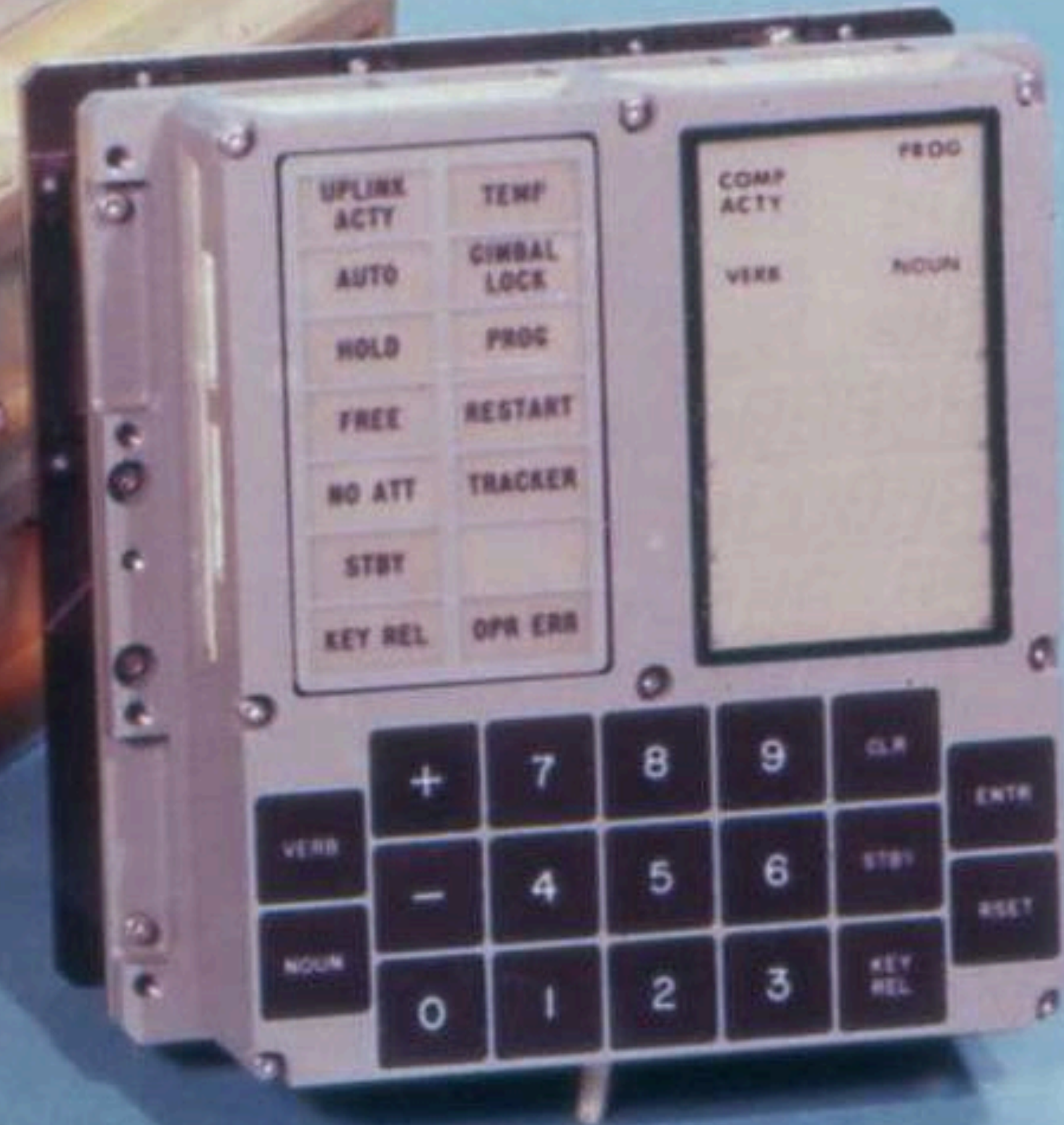




# Apollo Guidance Computer

MIT  
1966  
\$200,000  
57 built

1.024 MHz  
15 bit  
2 KW RAM  
36 KW ROM



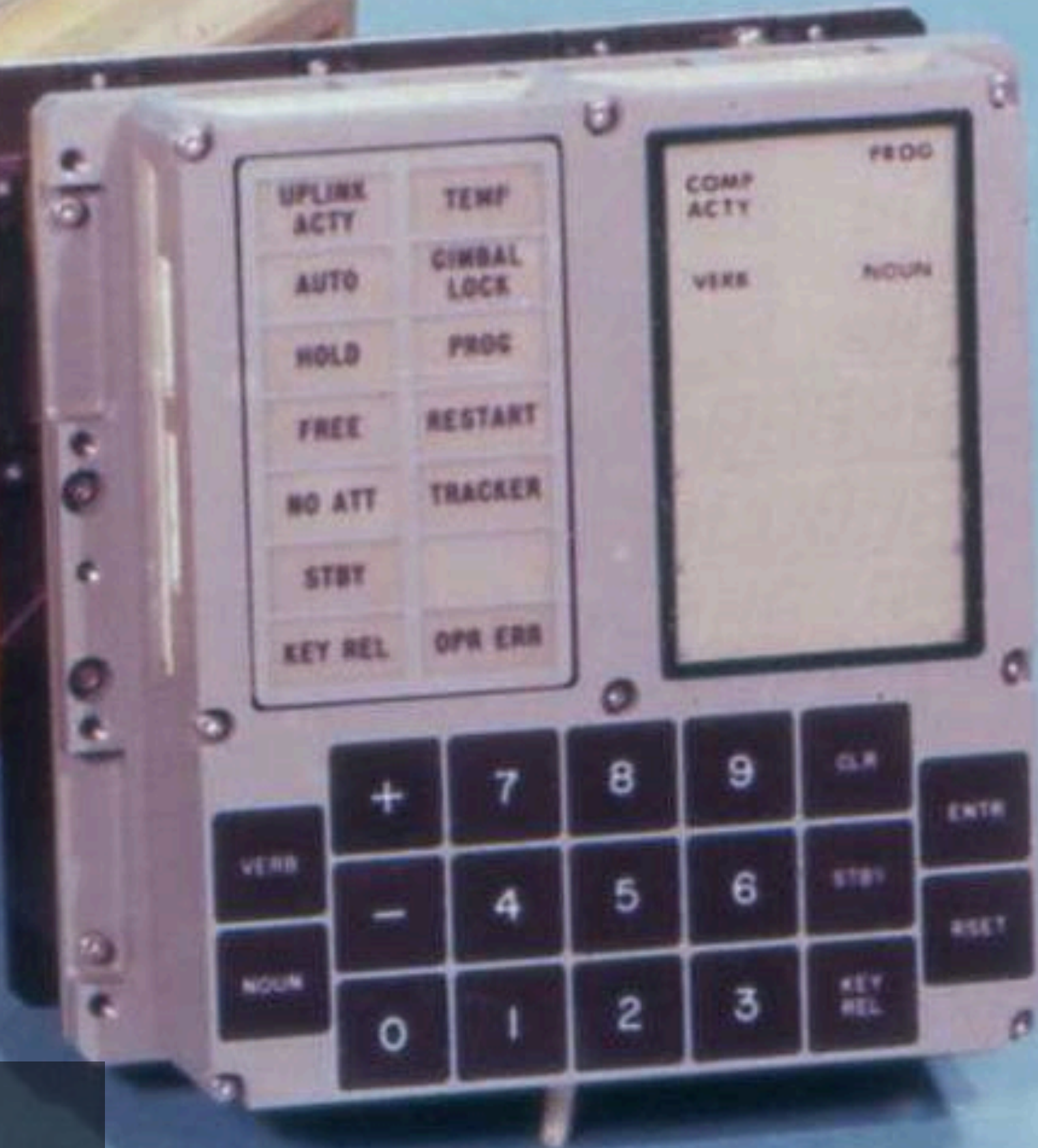


# Apollo Guidance Computer

MIT  
1966  
\$200,000  
57 built

1.024 MHz  
15 bit  
2 KW RAM  
36 KW ROM

55x33x15 cm  
26.3 kg  
55 W





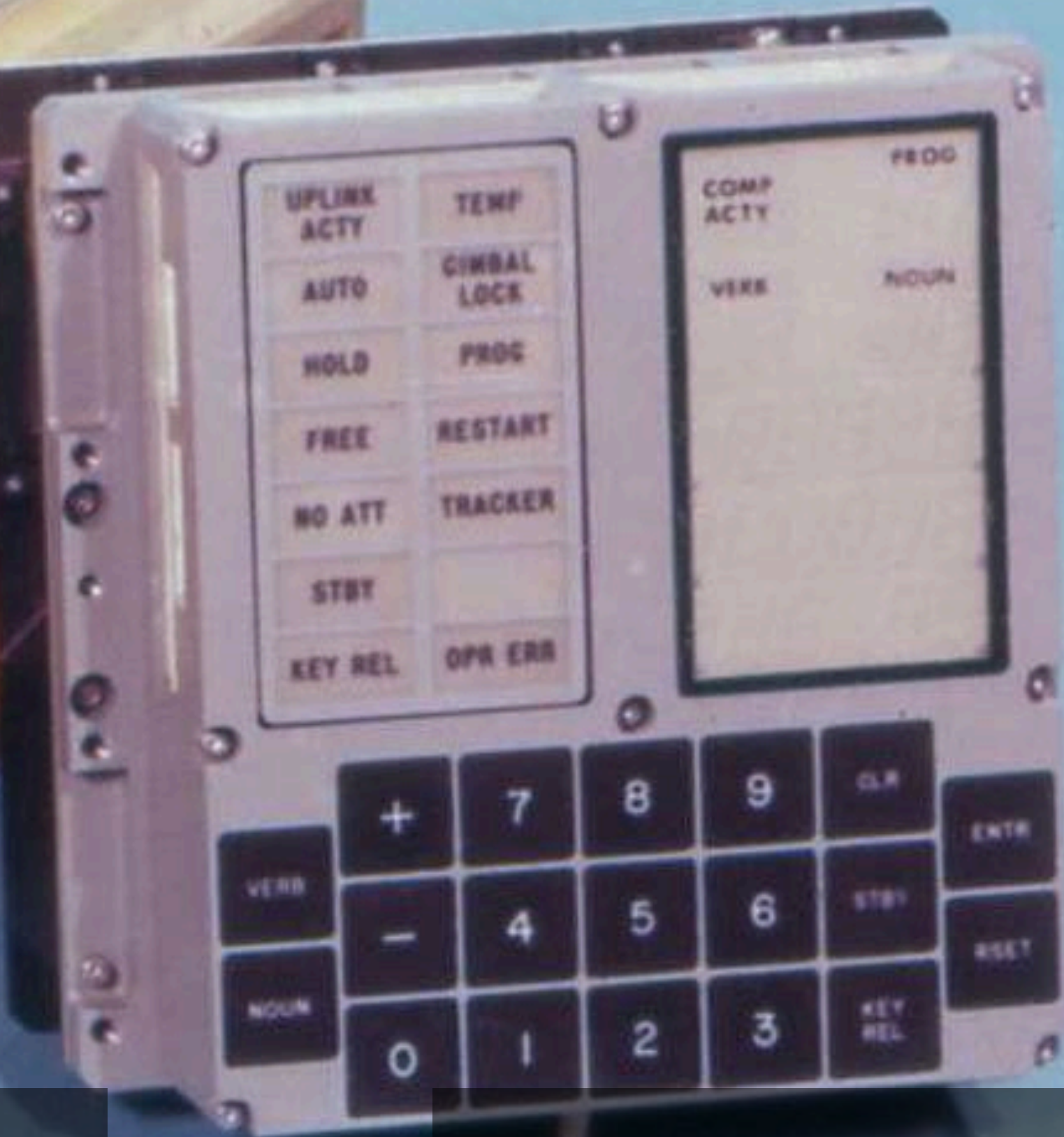
# Apollo Guidance Computer

MIT  
1966  
\$200,000  
57 built

1.024 MHz  
15 bit  
2 KW RAM  
36 KW ROM

55x33x15 cm  
26.3 kg  
55 W

numeric  
keyboard  
&  
display





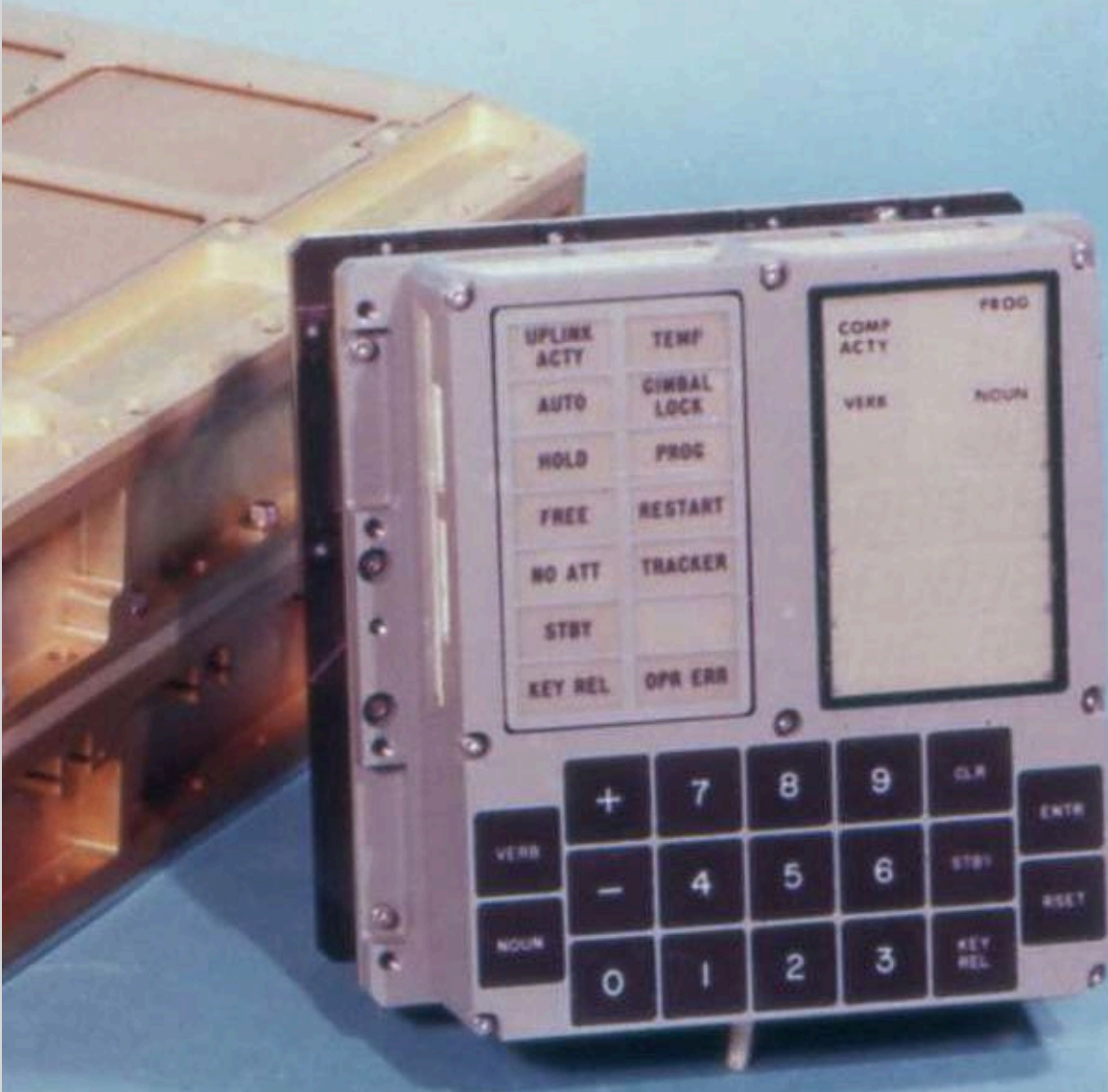
DEC PDP-8



1965

Transistors	
10,000	16,800
Memory Speed	
600 KHz	85 KHz
Word Size	
12 bit	15 bit
Kilo-Adds per Second	
384	44
Kilo-Muls per Second	
0.3	22
Weight	
82 kg	32 kg
Power Consumption	
400 W	55 W

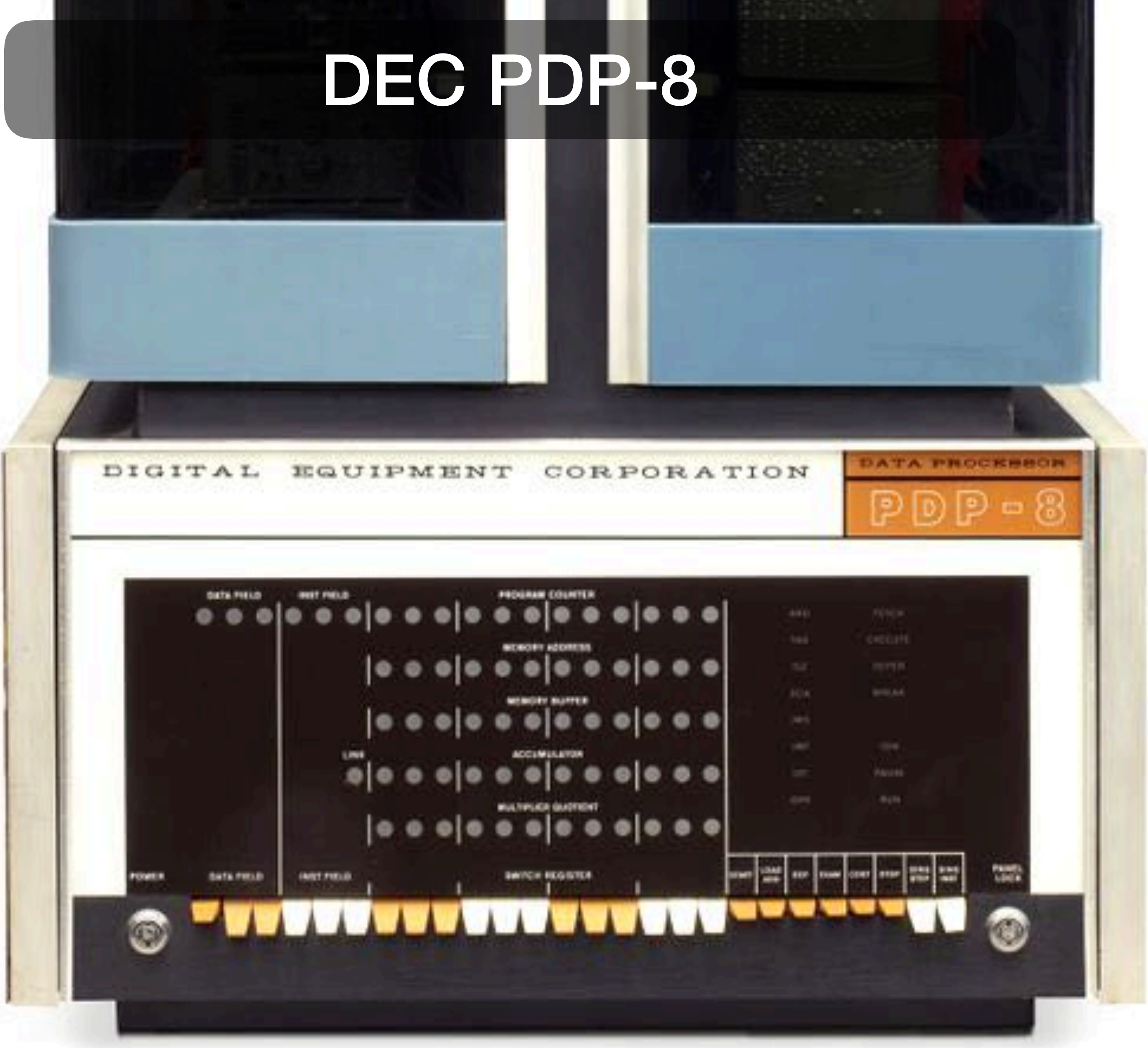
Apollo  
Guidance  
Computer



1966

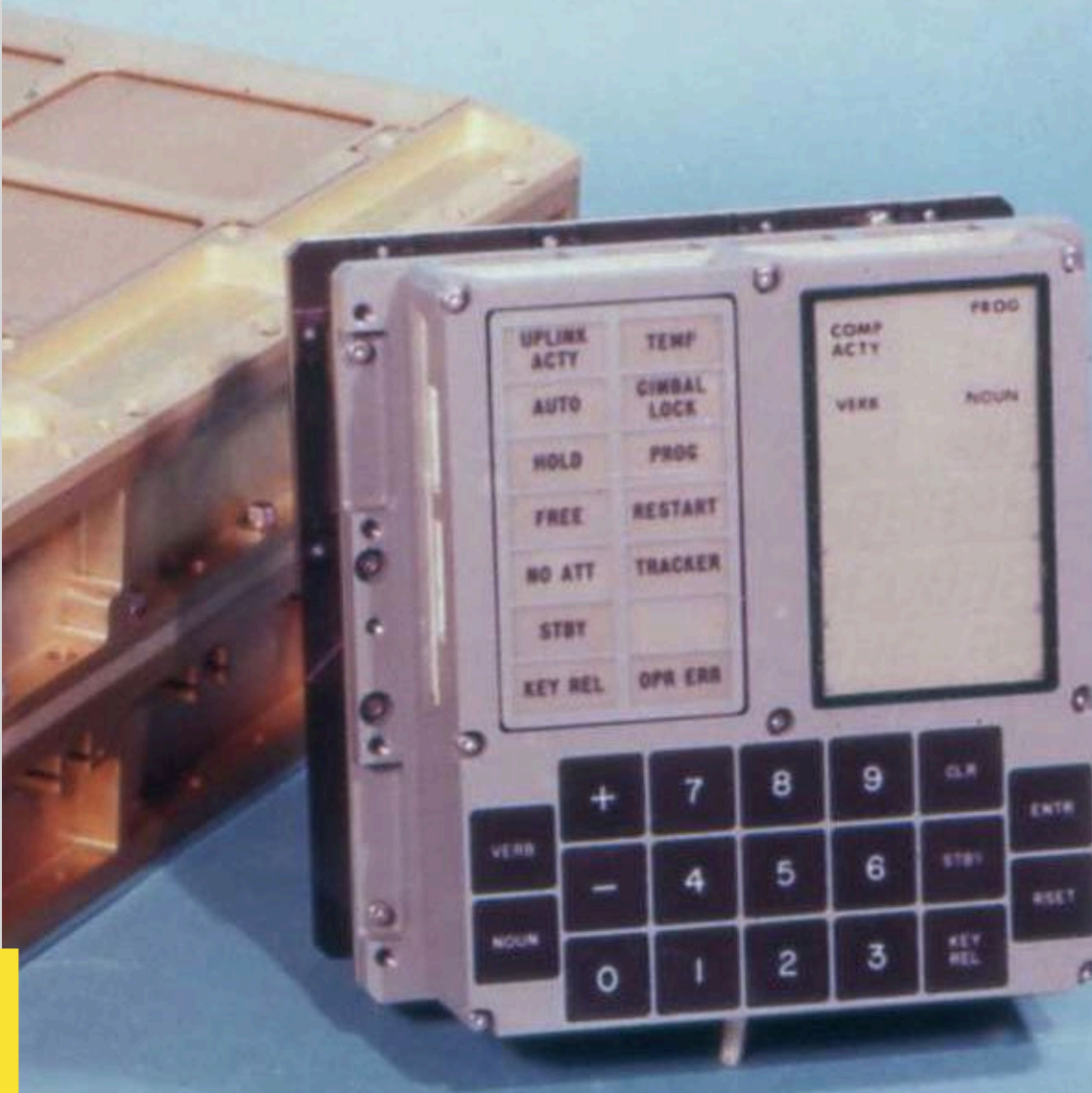


DEC PDP-8



1965

Apollo  
Guidance  
Computer



1966

Transistors	
10,000	16,800
Memory Speed	
600 KHz	85 KHz
Word Size	
12 bit	15 bit
Kilo-Adds per Second	
384	44
Kilo-Muls per Second	
0.3	22
Weight	
82 kg	32 kg
Power Consumption	
400 W	55 W

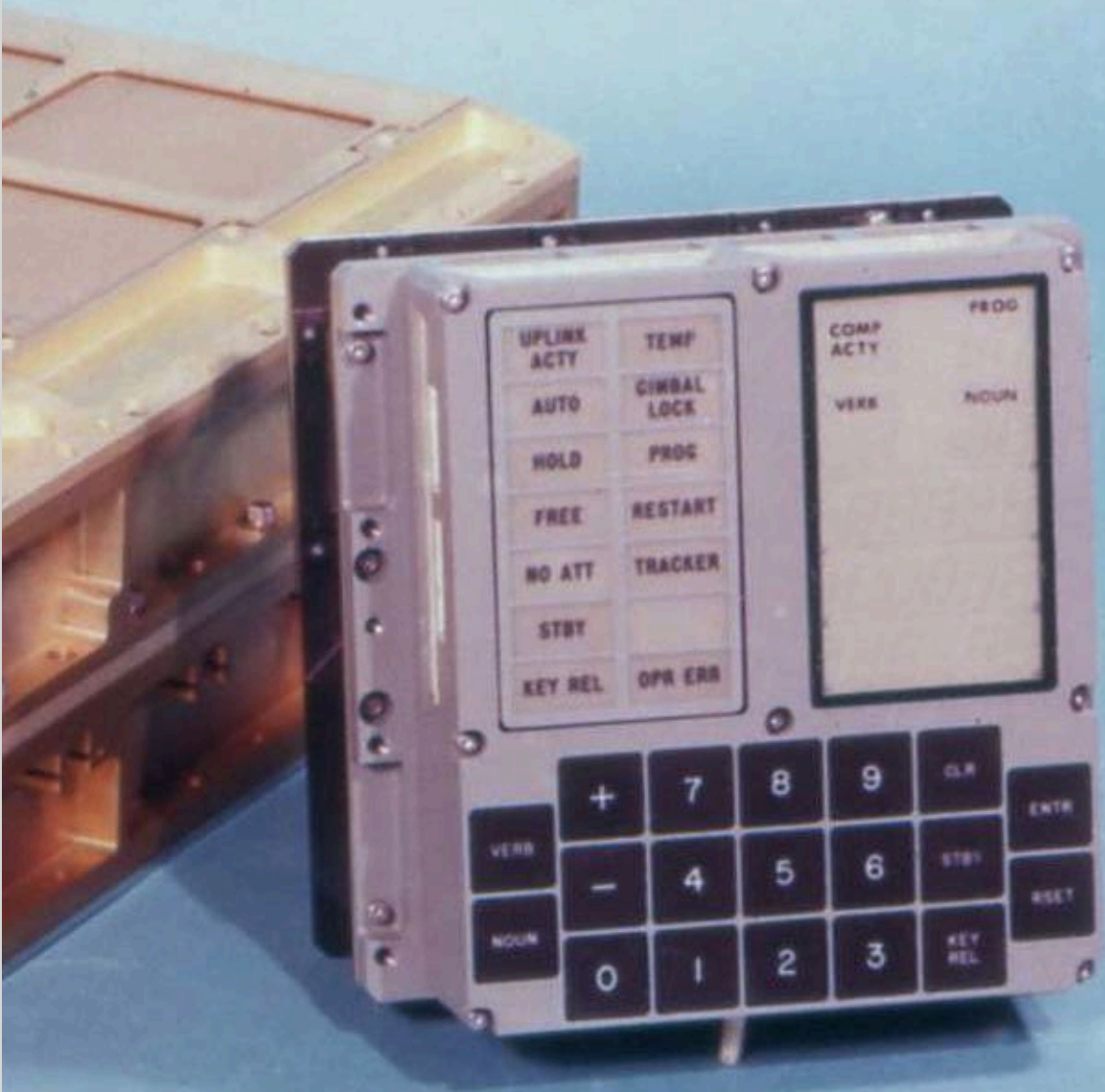


# UNIVAC I



1951

# Apollo Guidance Computer

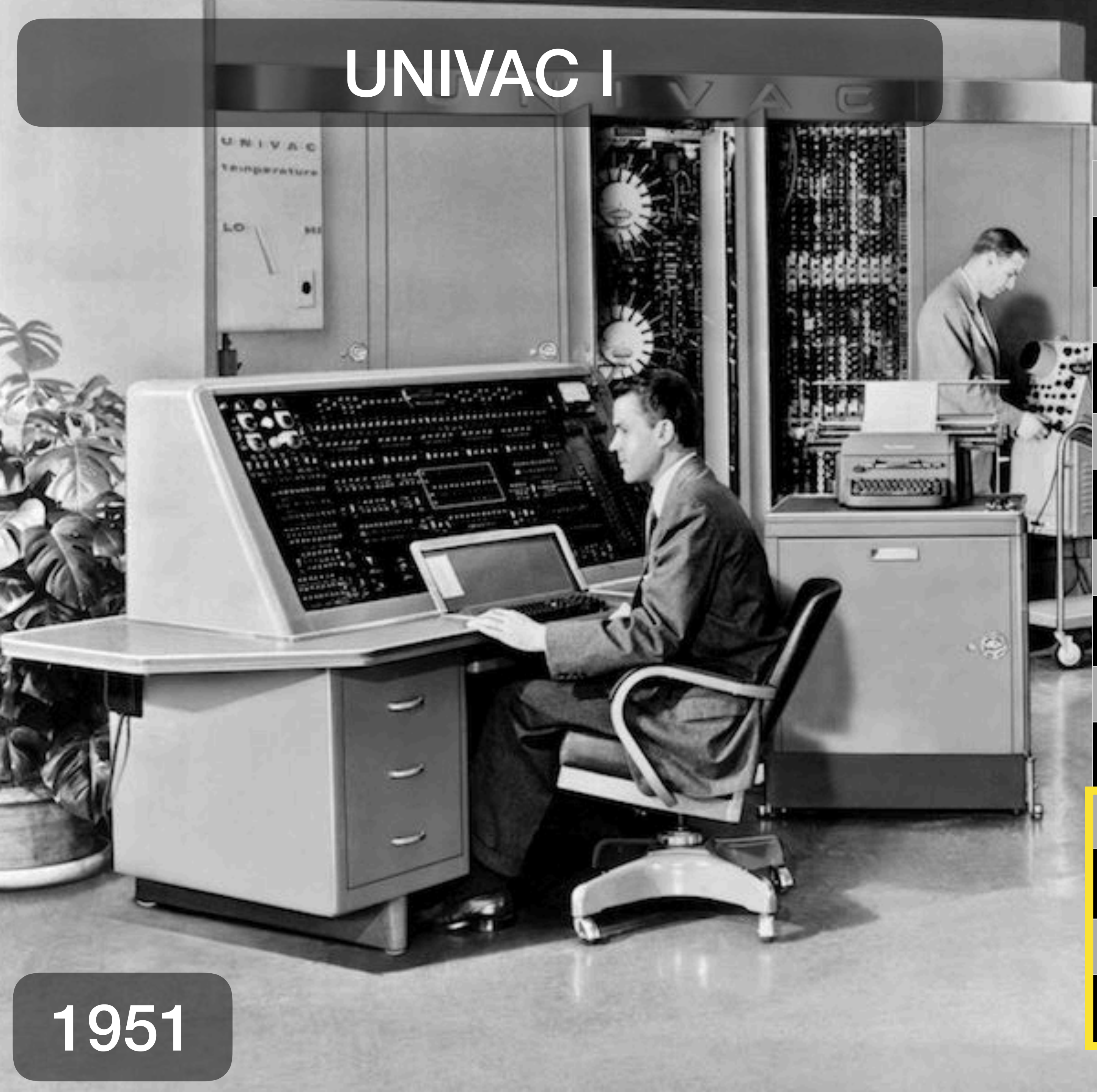


1966

Transistors	
5,000	16,800
Memory Speed	
0.5 KHz	85 KHz
Word Size	
48 bit	15 bit
Kilo-Adds per Second	
1.9	44
Kilo-Muls per Second	
0.5	22
Weight	
7,300 kg	32 kg
Power Consumption	
125,000 W	55 W



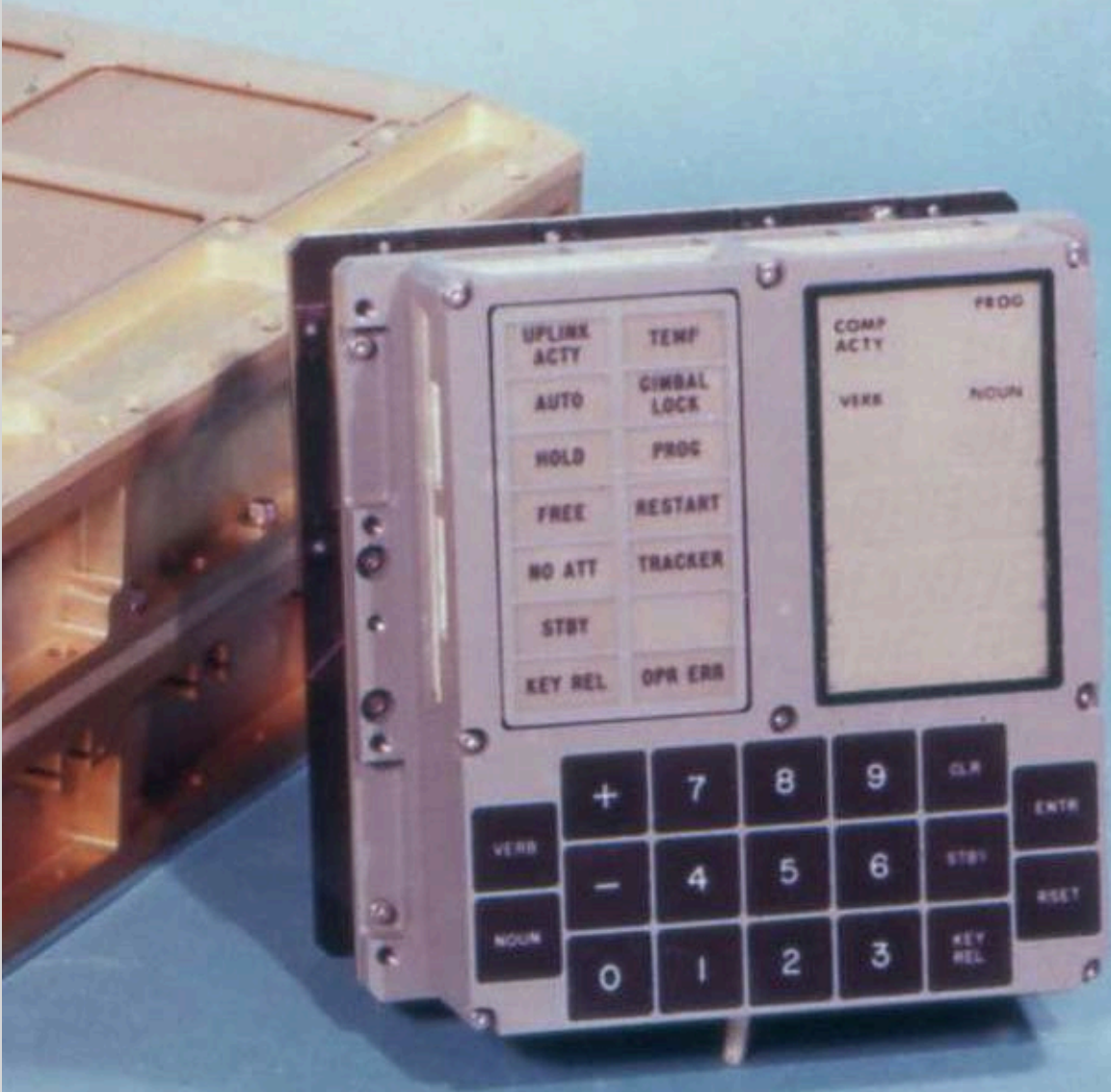
# UNIVAC I



1951

Transistors	
5,000	16,800
Memory Speed	
0.5 KHz	85 KHz
Word Size	
48 bit	15 bit
Kilo-Adds per Second	
1.9	44
Kilo-Muls per Second	
0.5	22
Weight	
7,300 kg	32 kg
Power Consumption	
125,000 W	55 W

# Apollo Guidance Computer



1966



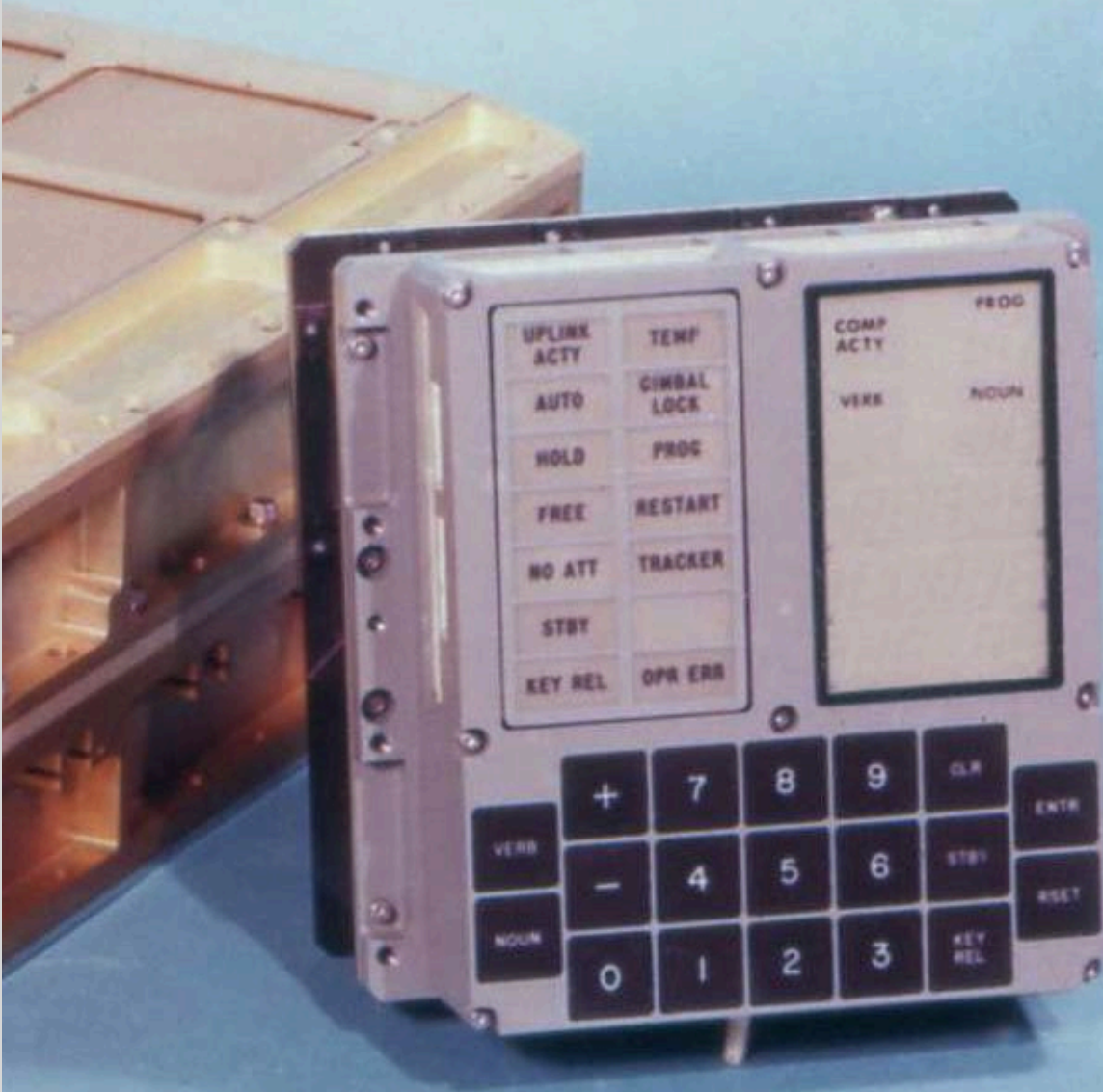
# Commodore PET



1977

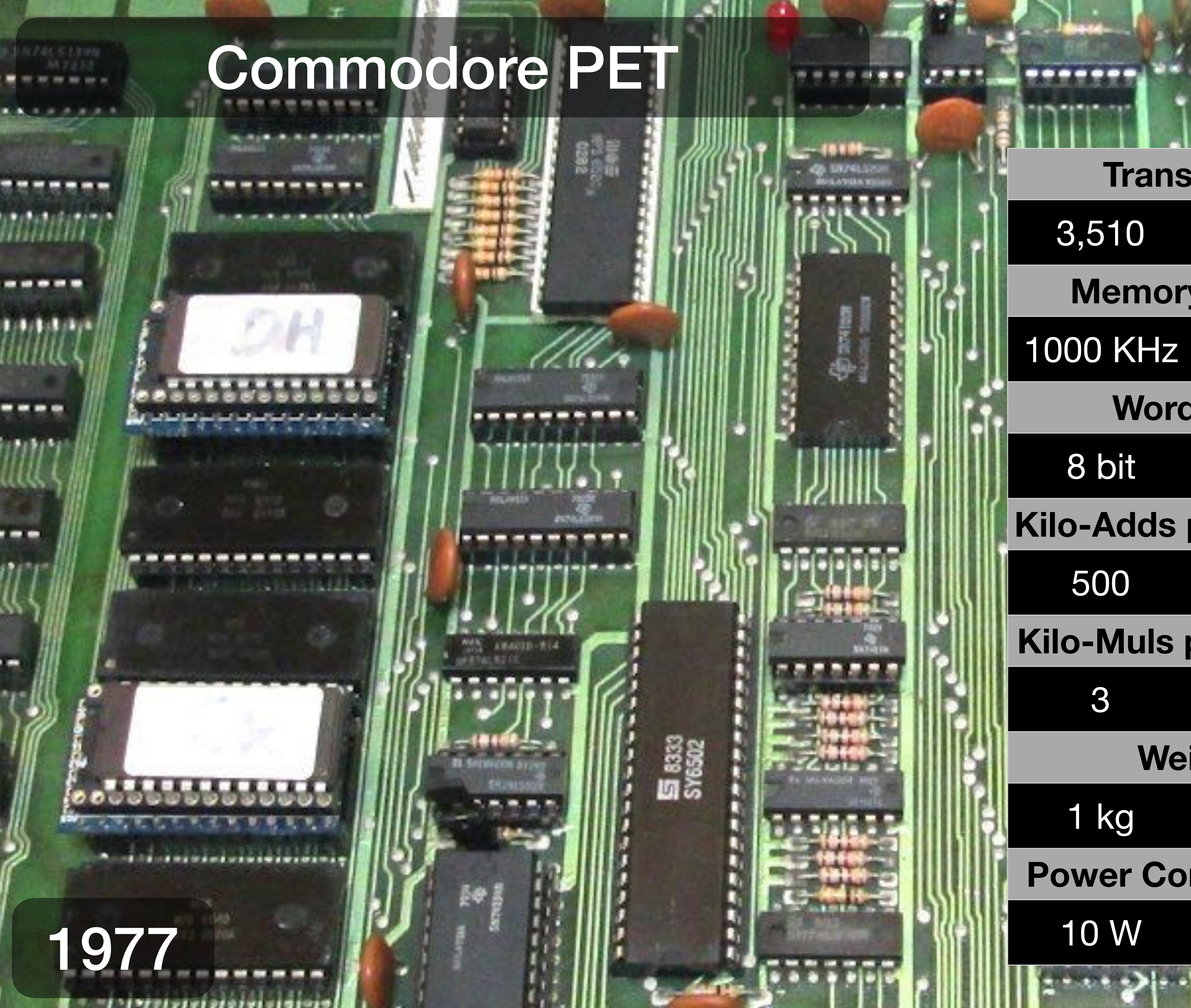
Transistors	
3,510	16,800
Memory Speed	
1000 KHz	85 KHz
Word Size	
8 bit	15 bit
Kilo-Adds per Second	
500	44
Kilo-Muls per Second	
3	22
Weight	
1 kg	32 kg
Power Consumption	
10 W	55 W

# Apollo Guidance Computer



1966

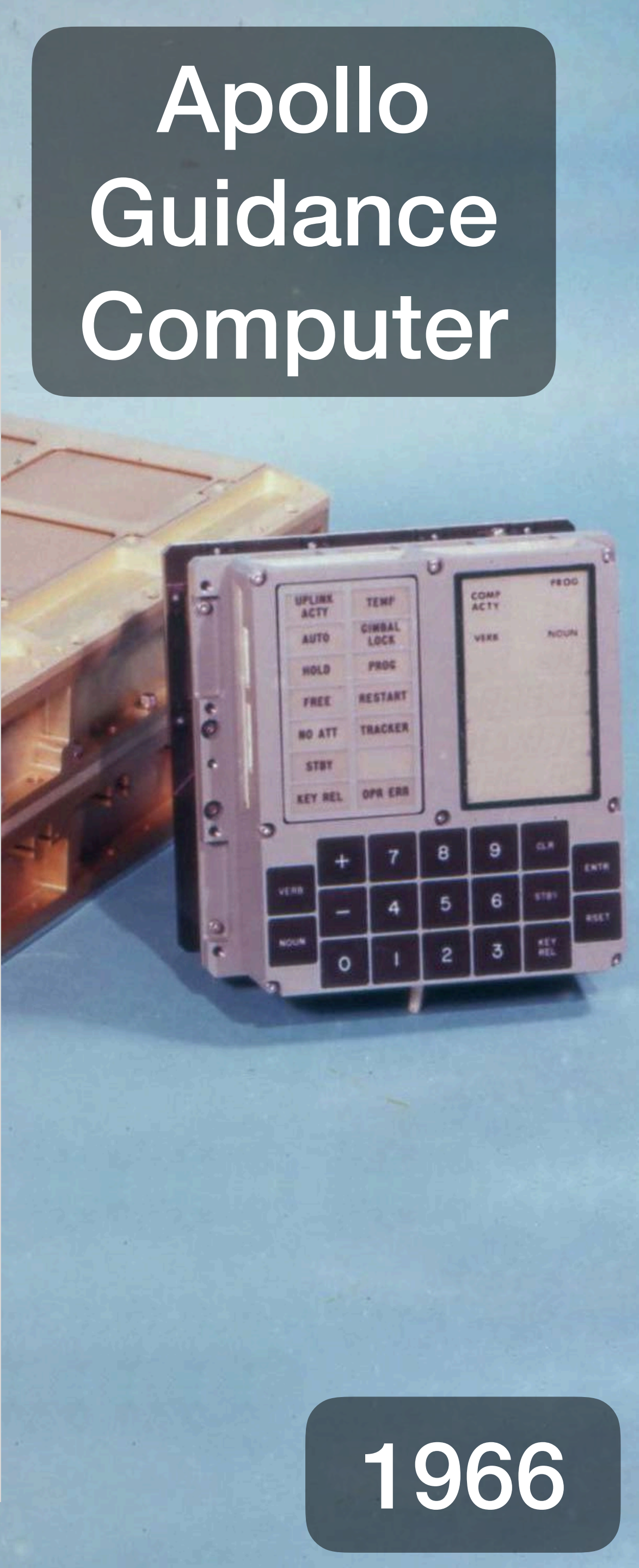




# Commodore PET

1977

Transistors	
3,510	16,800
Memory Speed	
1000 KHz	85 KHz
Word Size	
8 bit	15 bit
Kilo-Adds per Second	
500	44
Kilo-Muls per Second	
3	22
Weight	
1 kg	32 kg
Power Consumption	
10 W	55 W



# Apollo Guidance Computer

1966







**15 bit**

**One's  
Complement**

**Accumulator  
Machine**

**very 60s  
very alien**



**15 bit**

**One's  
Complement**

**Accumulator  
Machine**

**very 60s  
very alien**

**Integrated  
Circuits**

**Microcoded**

**Core Rope  
Memory**

**innovative  
for its time**



**15 bit**

**One's  
Complement**

**Accumulator  
Machine**

**very 60s  
very alien**

**Integrated  
Circuits**

**Microcoded**

**Core Rope  
Memory**

**innovative  
for its time**

**Gyroscope  
Accelerometer**

**Radar**

**Jets**

**very interesting  
peripherals**



**15 bit**

**One's  
Complement**

**Accumulator  
Machine**

**very 60s  
very alien**

**Integrated  
Circuits**

**Microcoded**

**Core Rope  
Memory**

**innovative  
for its time**

**Gyroscope  
Accelerometer**

**Radar**

**Jets**

**very interesting  
peripherals**

**Preemptive  
Real-Time**

**Fault-Tolerant**

**Virtual  
Machines**

**revolutionary  
for its time**



**15 bit**

**One's  
Complement**

**Accumulator  
Machine**

**very 60s  
very alien**

**Integrated  
Circuits**

**Microcoded**

**Core Rope  
Memory**

**innovative  
for its time**

**Gyroscope  
Accelerometer**

**Radar**

**Jets**

**very interesting  
peripherals**

**Preemptive  
Real-Time**

**Fault-Tolerant**

**Virtual  
Machines**

**revolutionary  
for its time**

**Guidance**

**Navigation**

**Fly-By-Wire**

**can fly you  
to the moon**









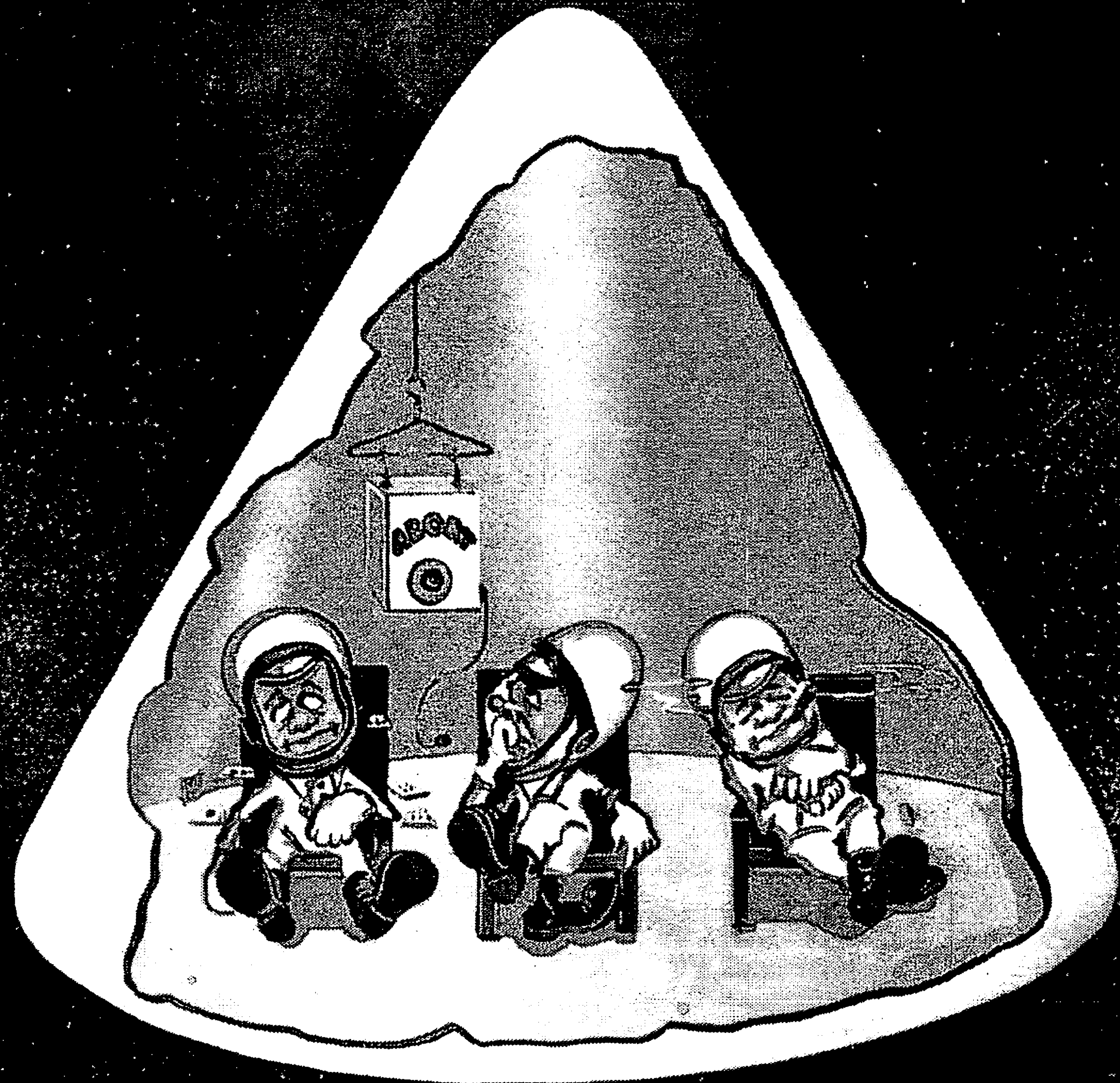




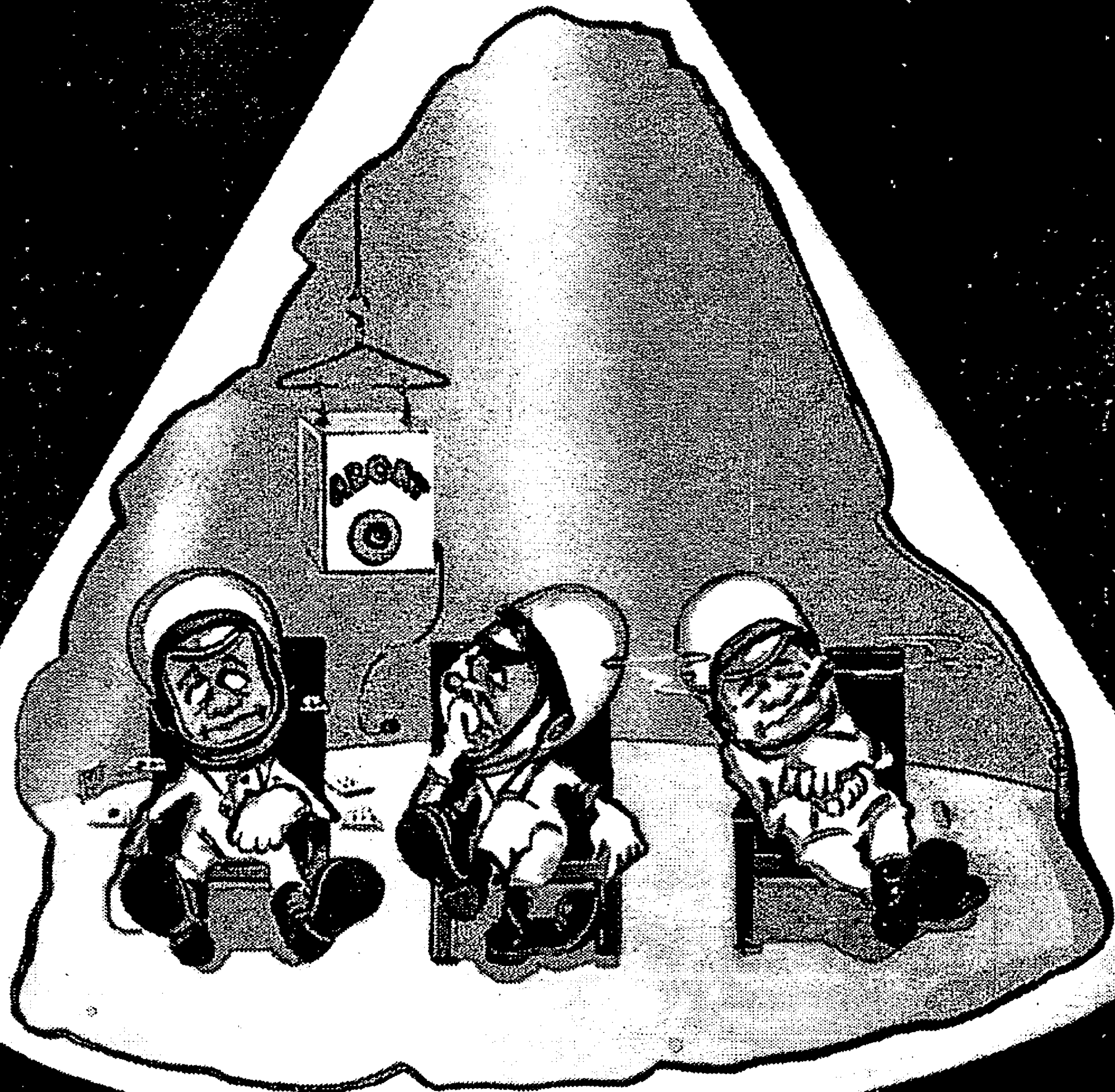




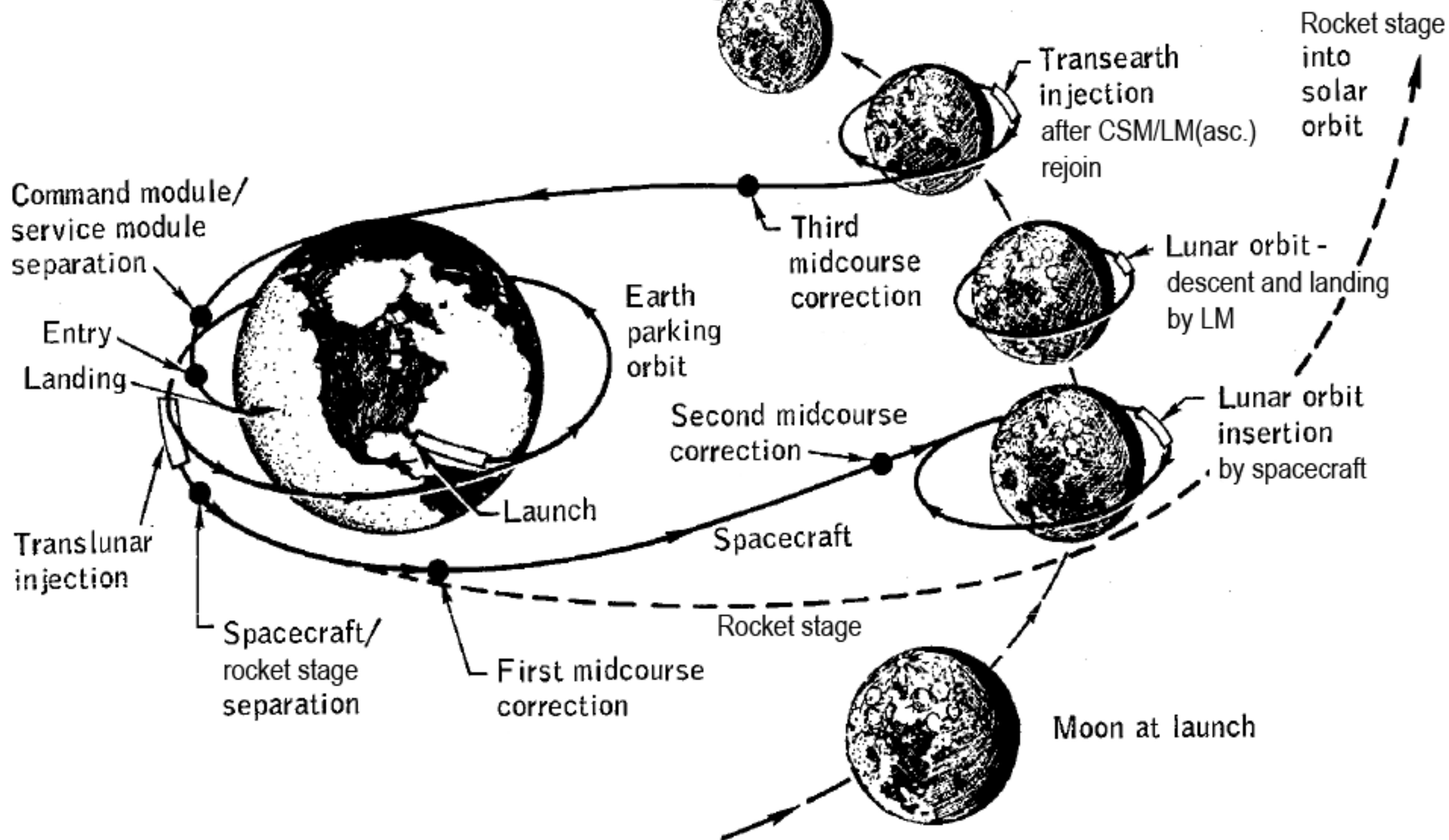






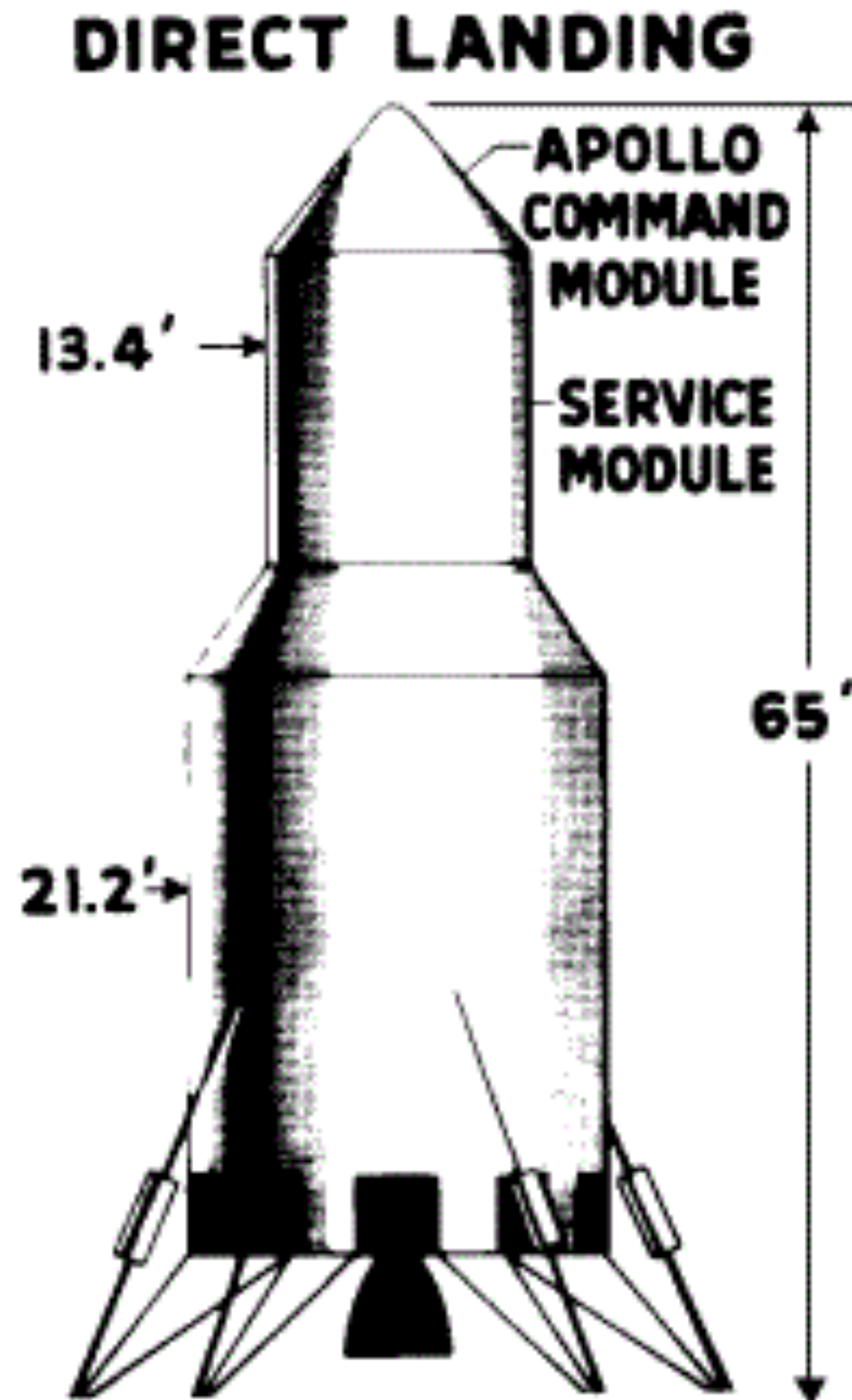








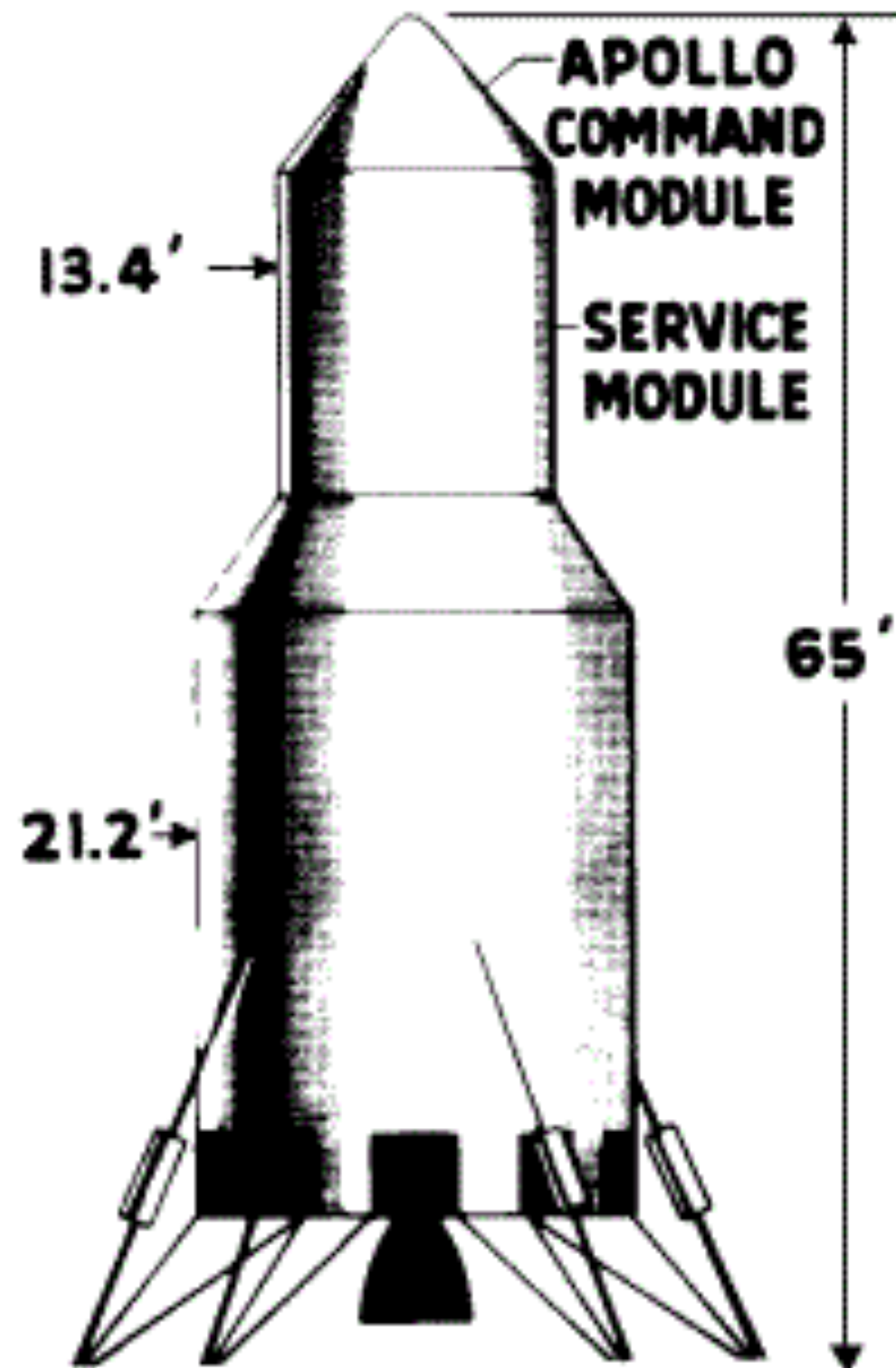
# Mission





# Mission

## DIRECT LANDING



## LUNAR FERRY OF LUNAR RENDEZVOUS

### LUNAR EXCURSION VEHICLE





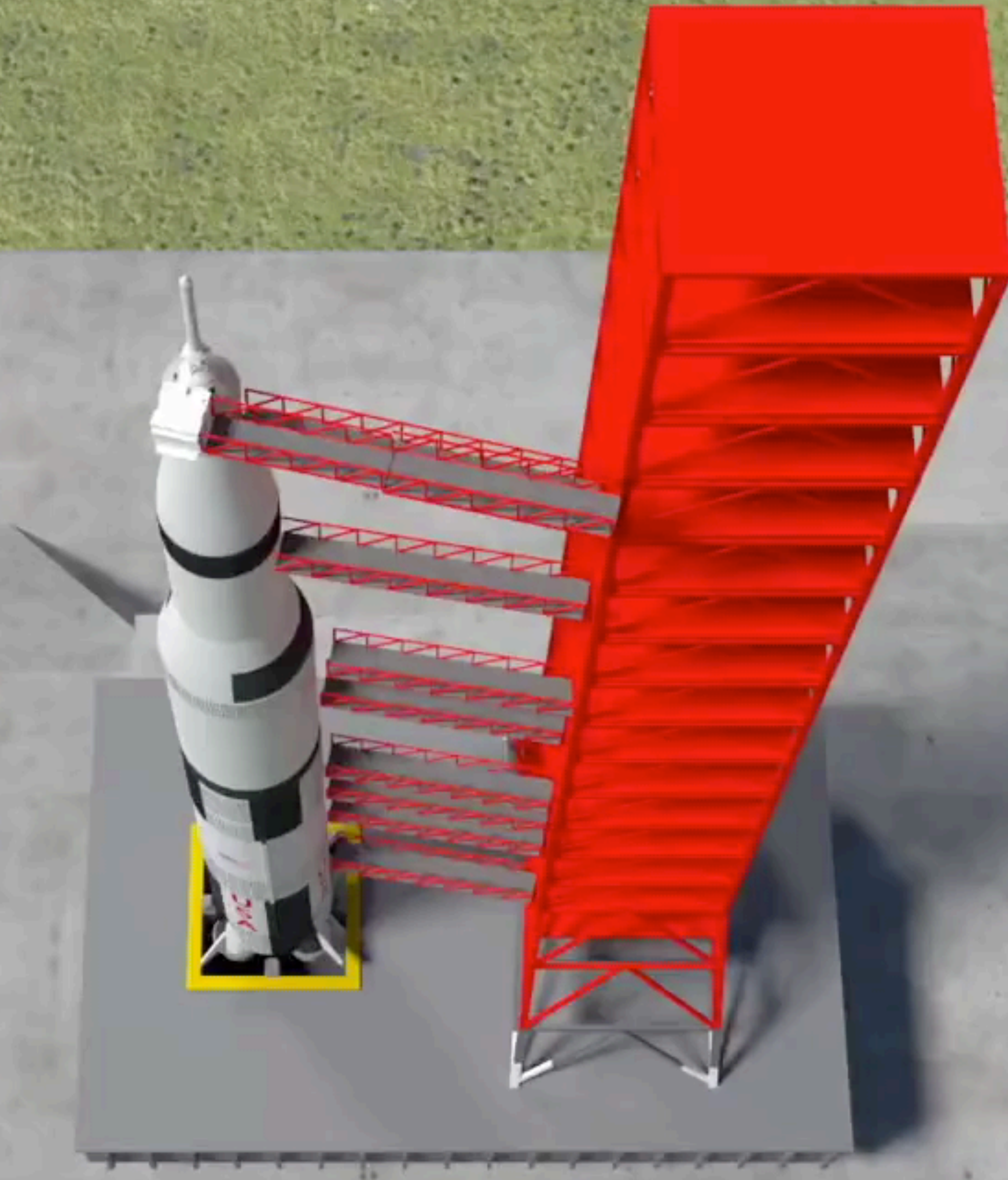
# Apollo Spacecraft



# Apollo Spacecraft

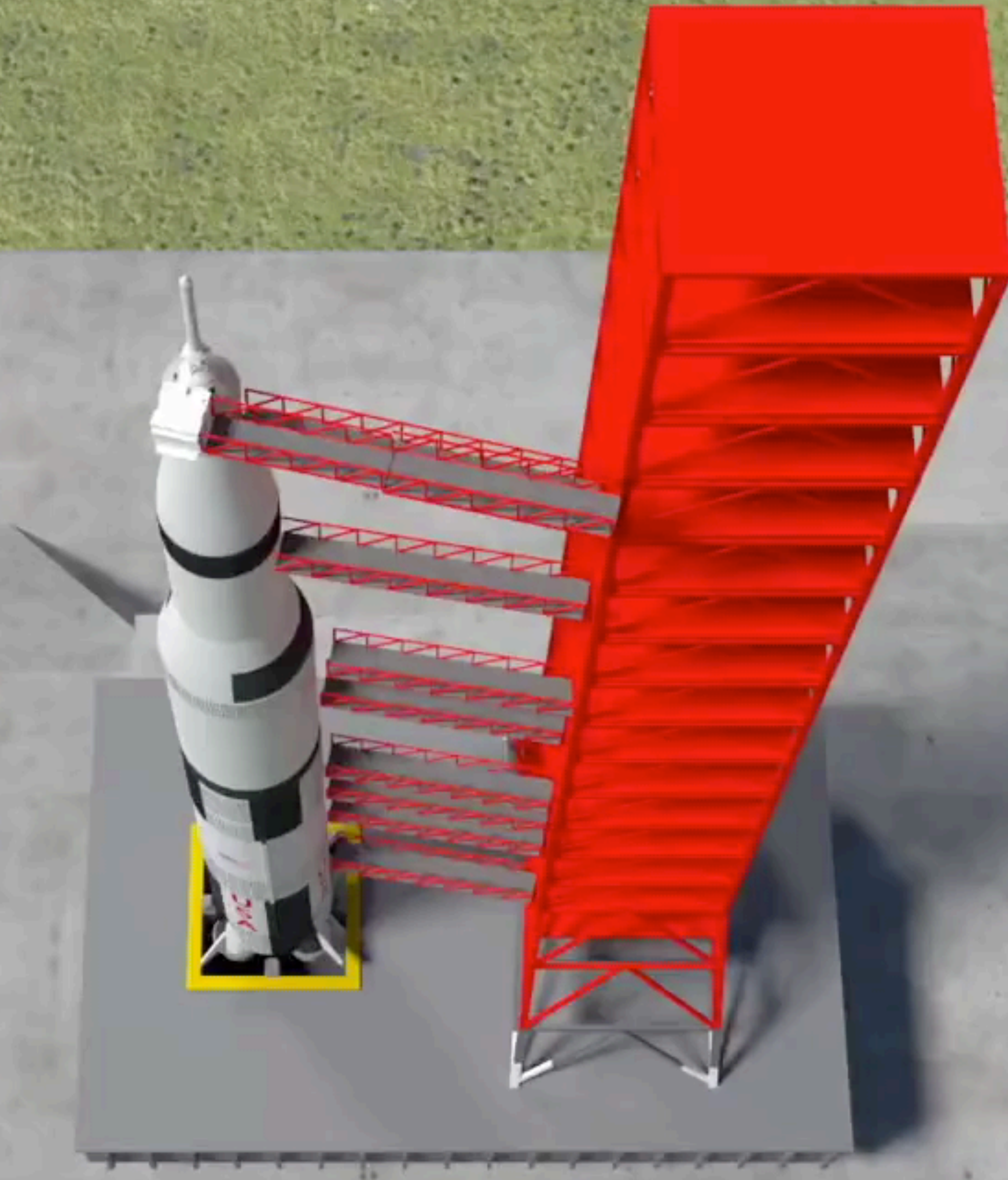


# Saturn V



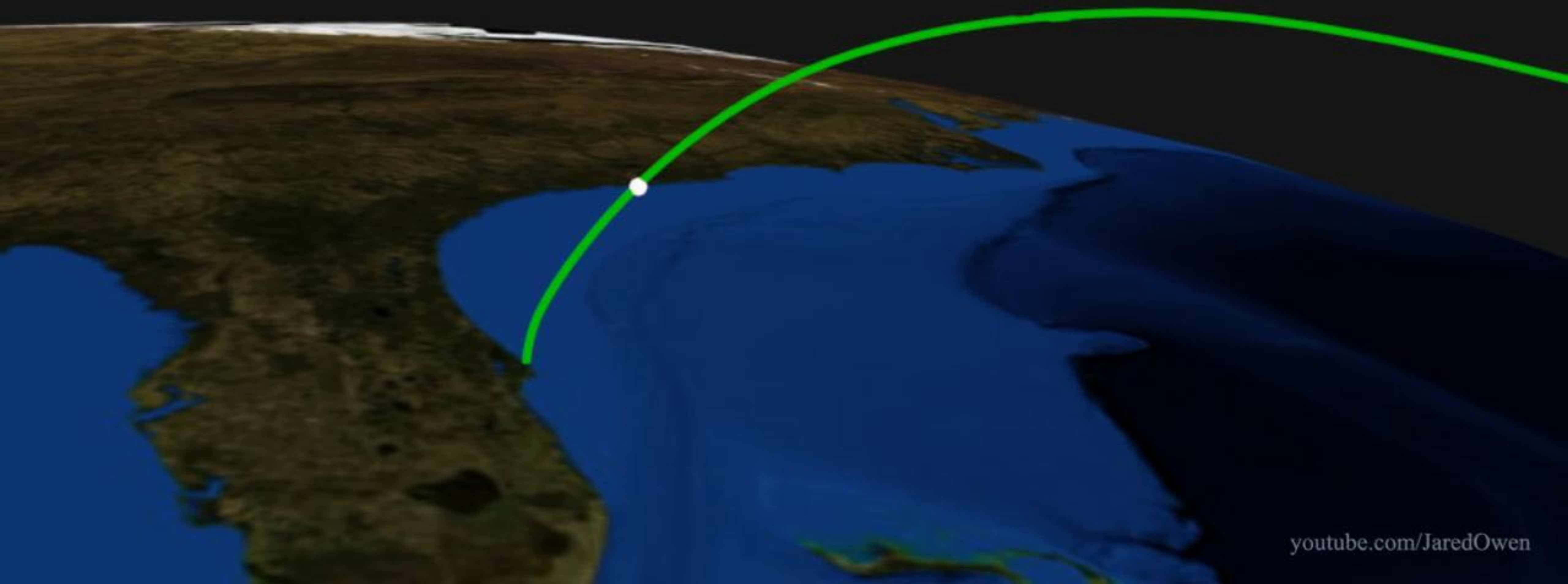


# Saturn V





# Launch into Orbit



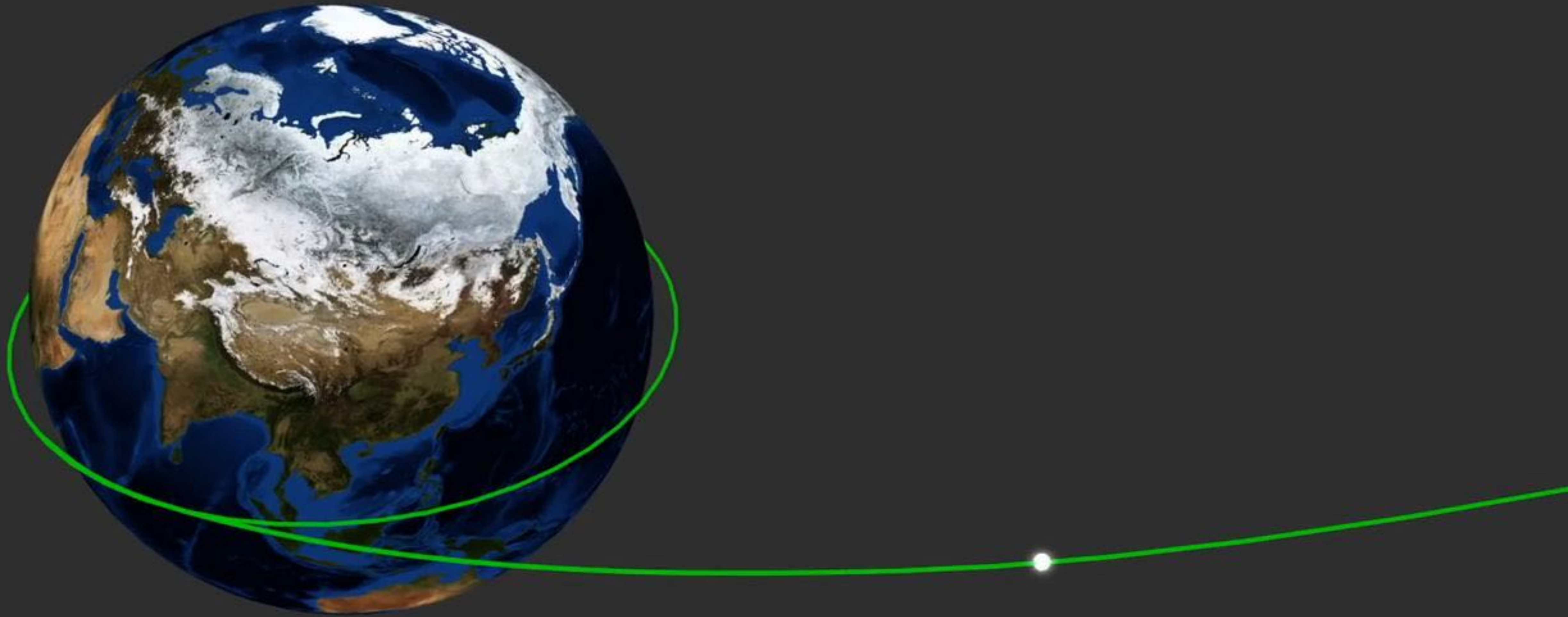


# Earth Orbit



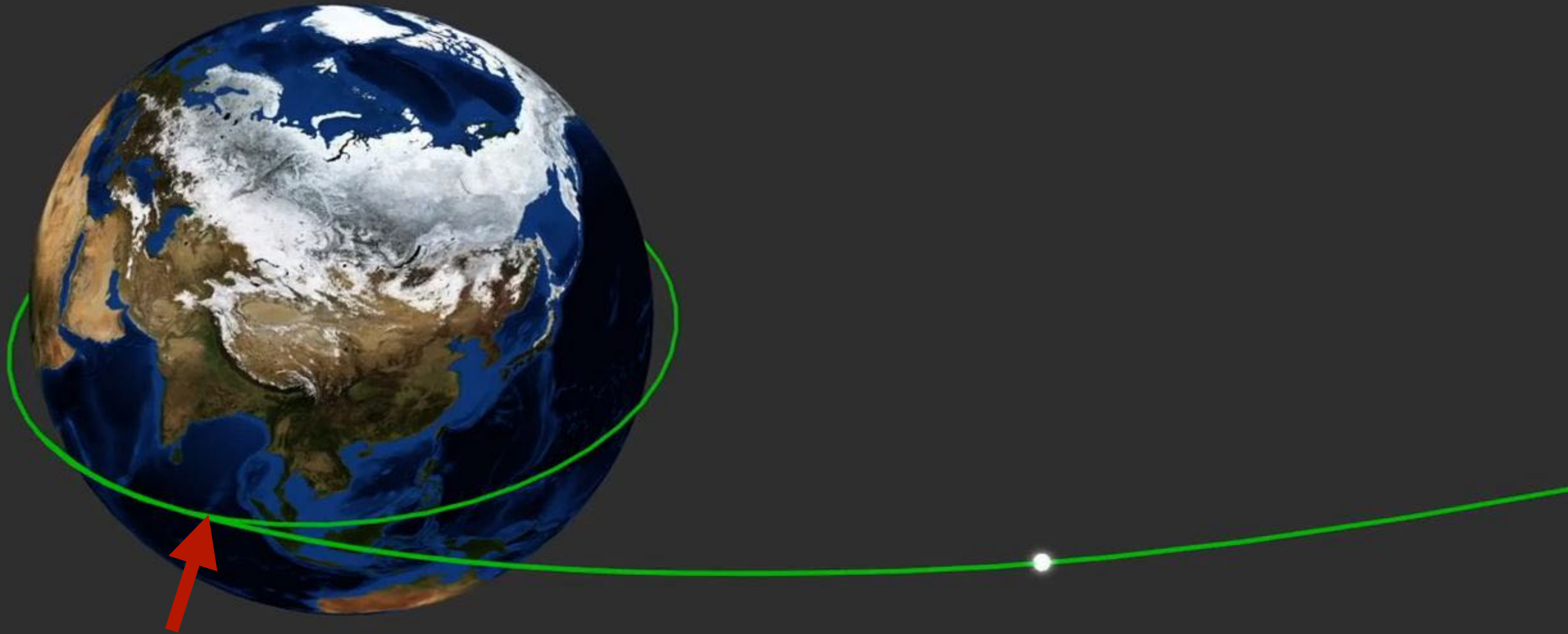


# Trans-Lunar Injection



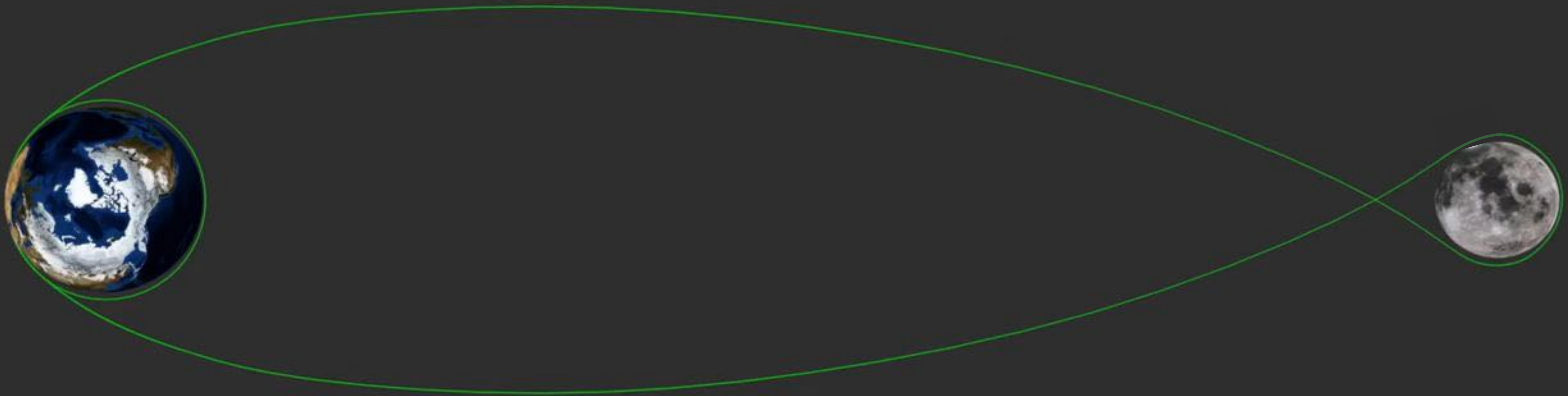


# Trans-Lunar Injection



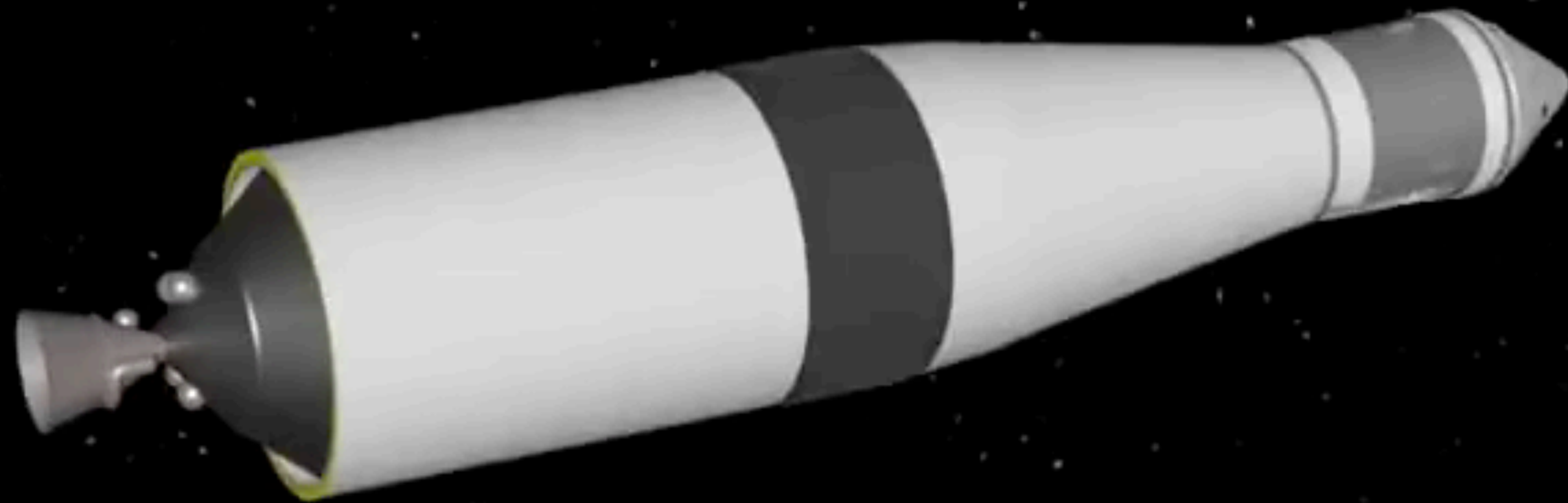


## Free Return Orbit



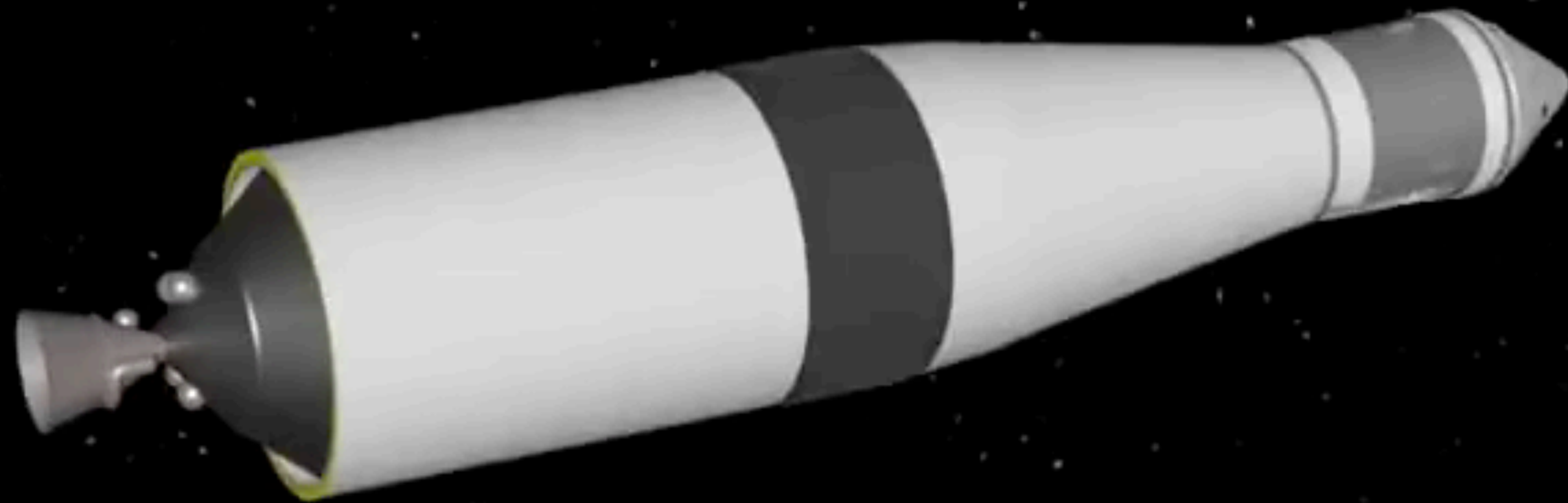


# Transposition & Docking





# Transposition & Docking





# Lunar Orbit Insertion





# Lunar Orbit Insertion



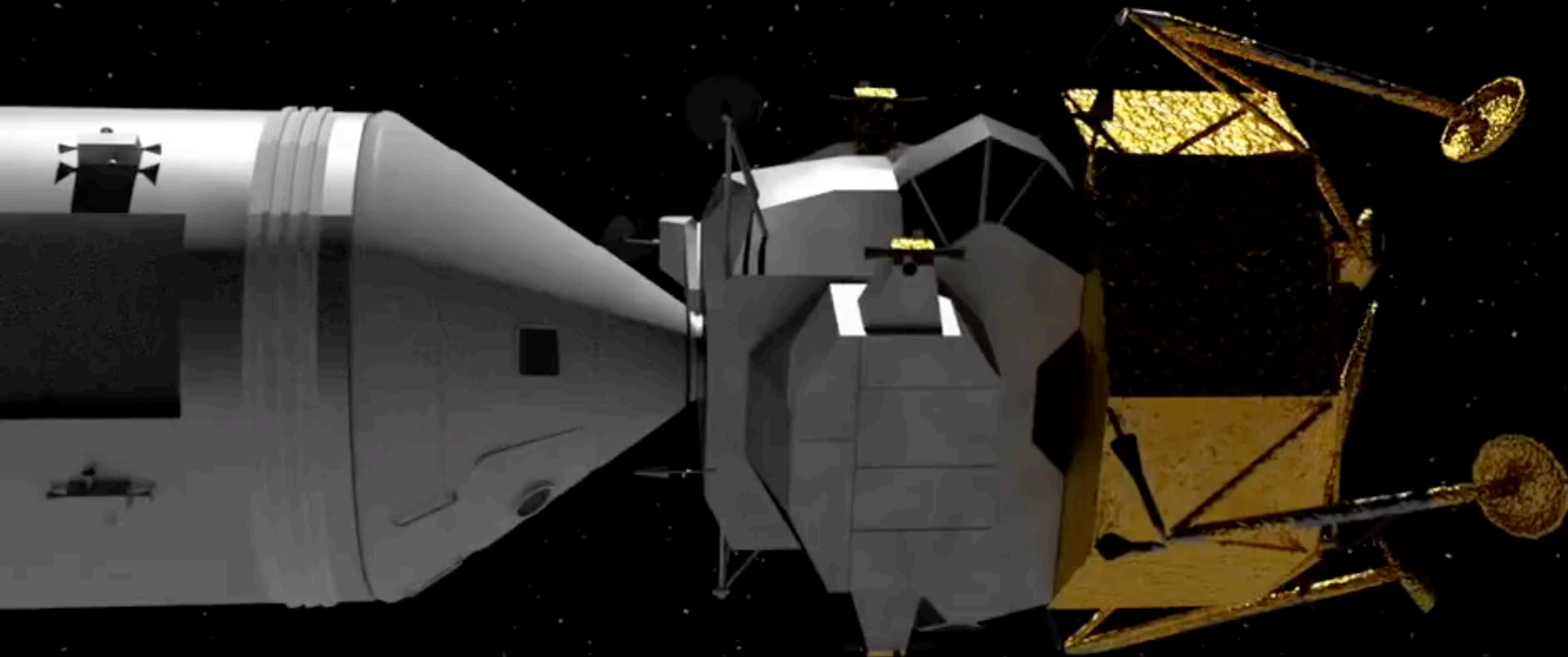


# Lunar Orbit Insertion



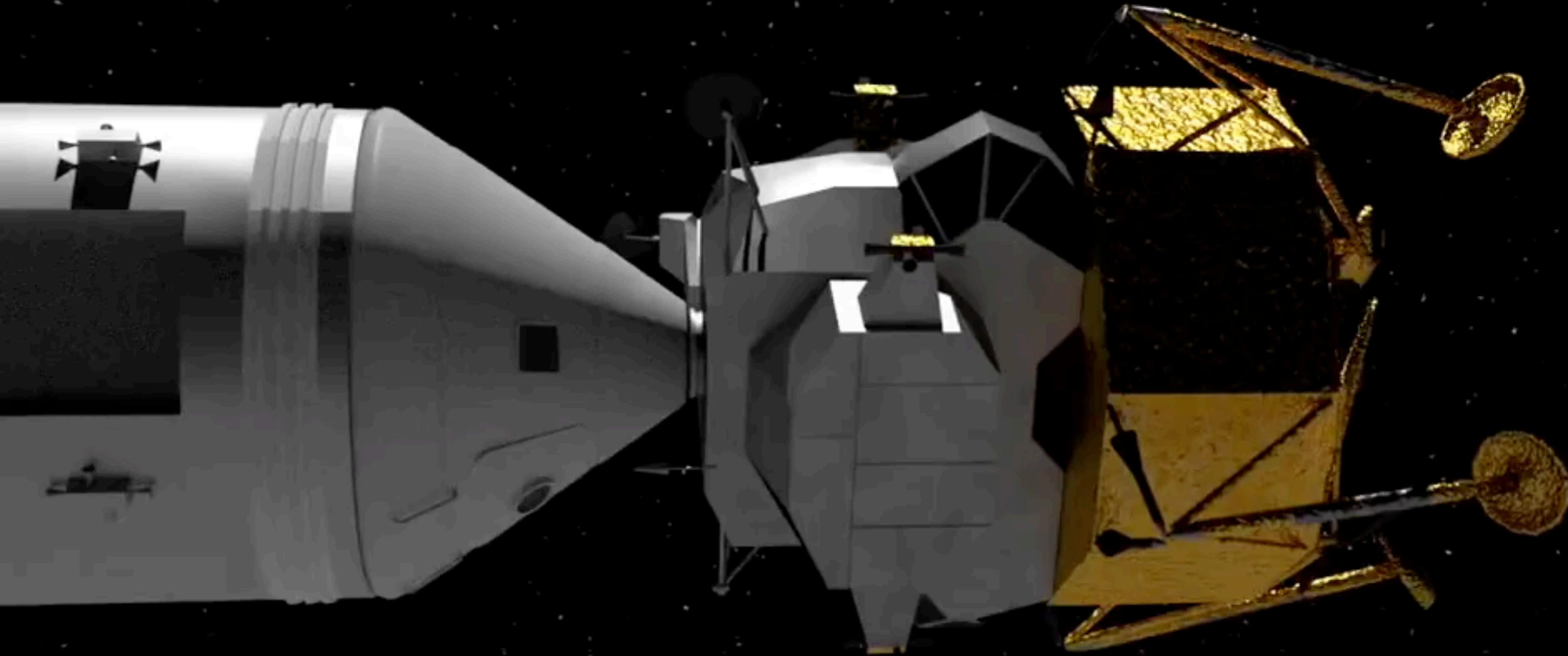


# Undocking





# Undocking



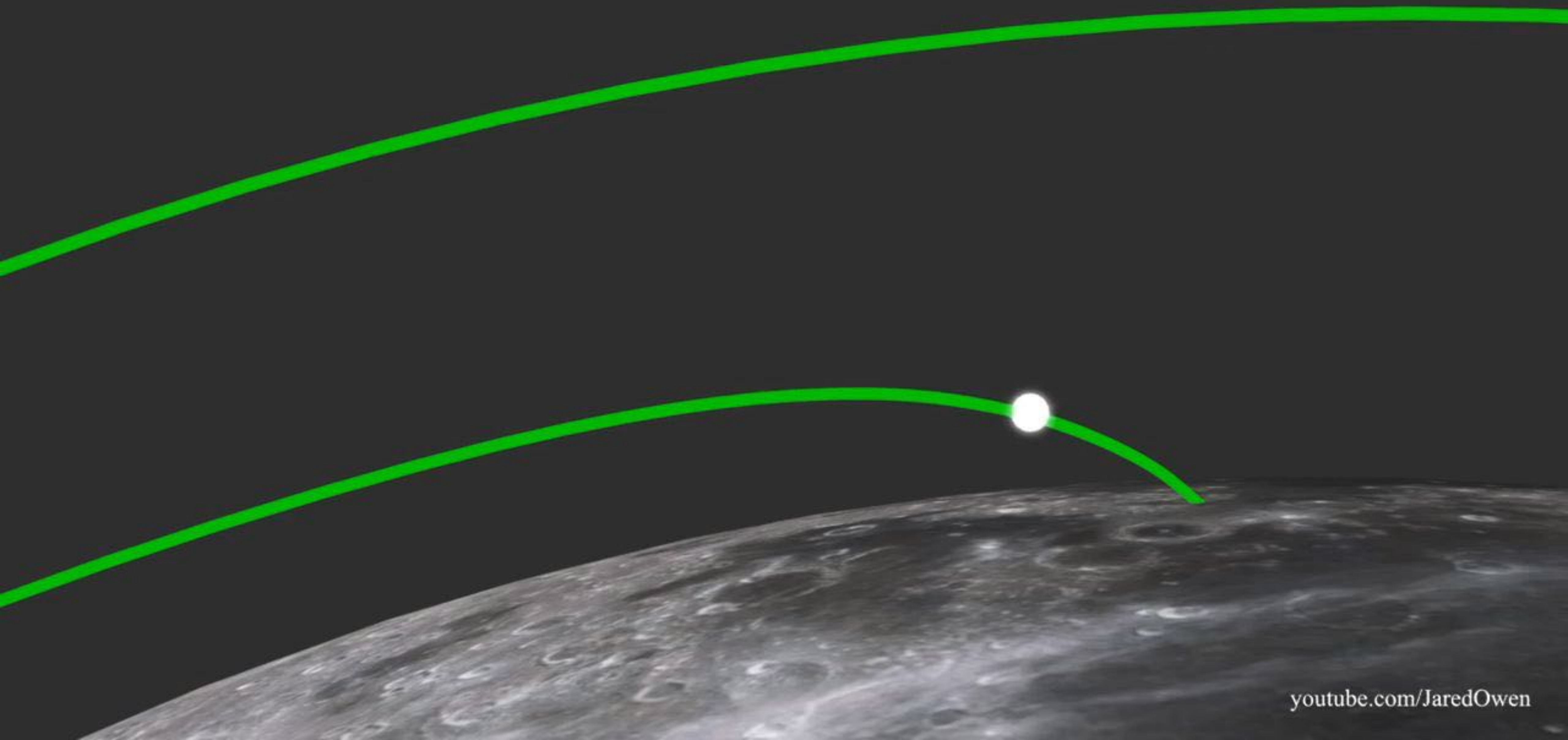


## Powered Descent



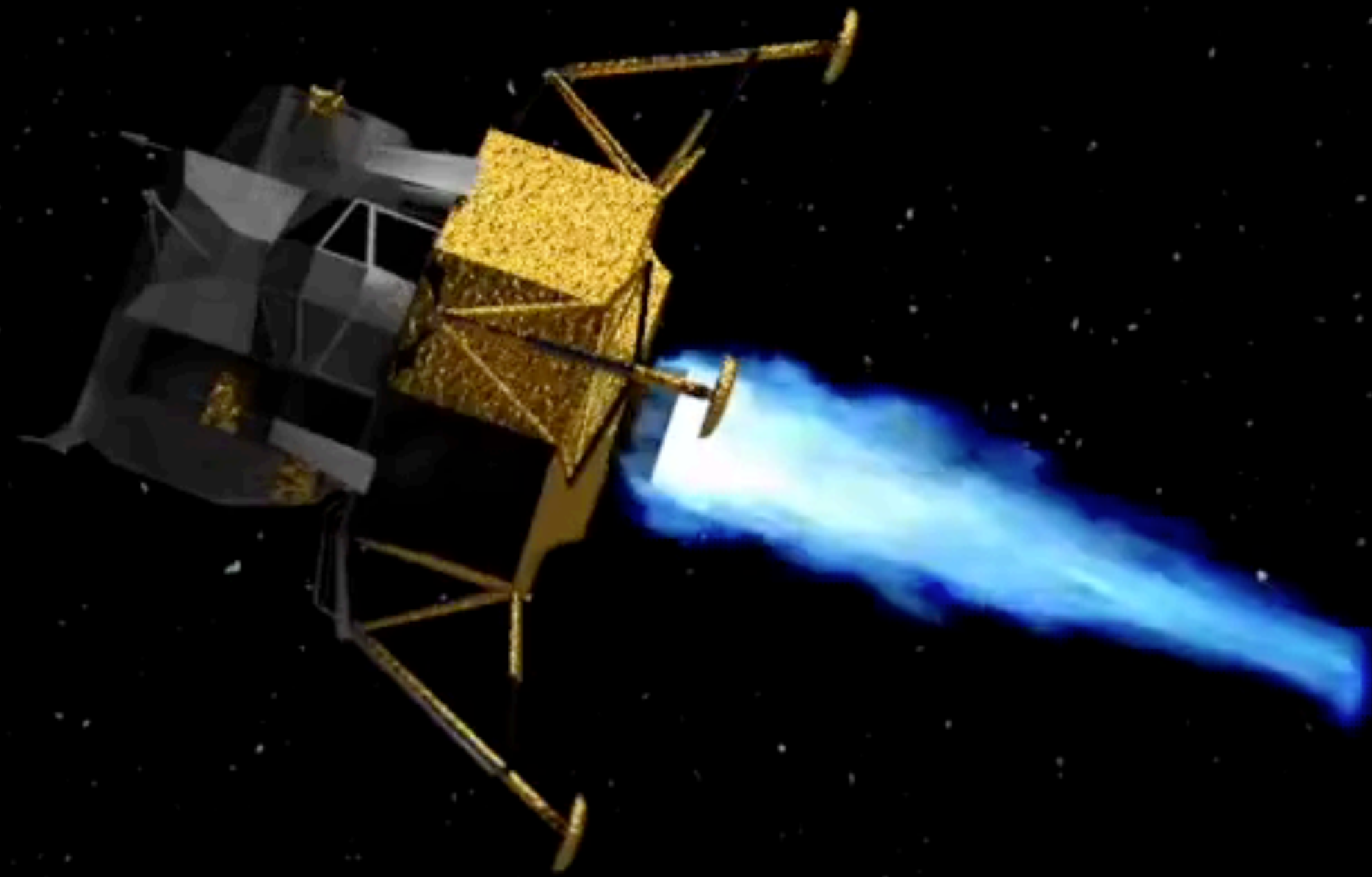


# Powered Descent



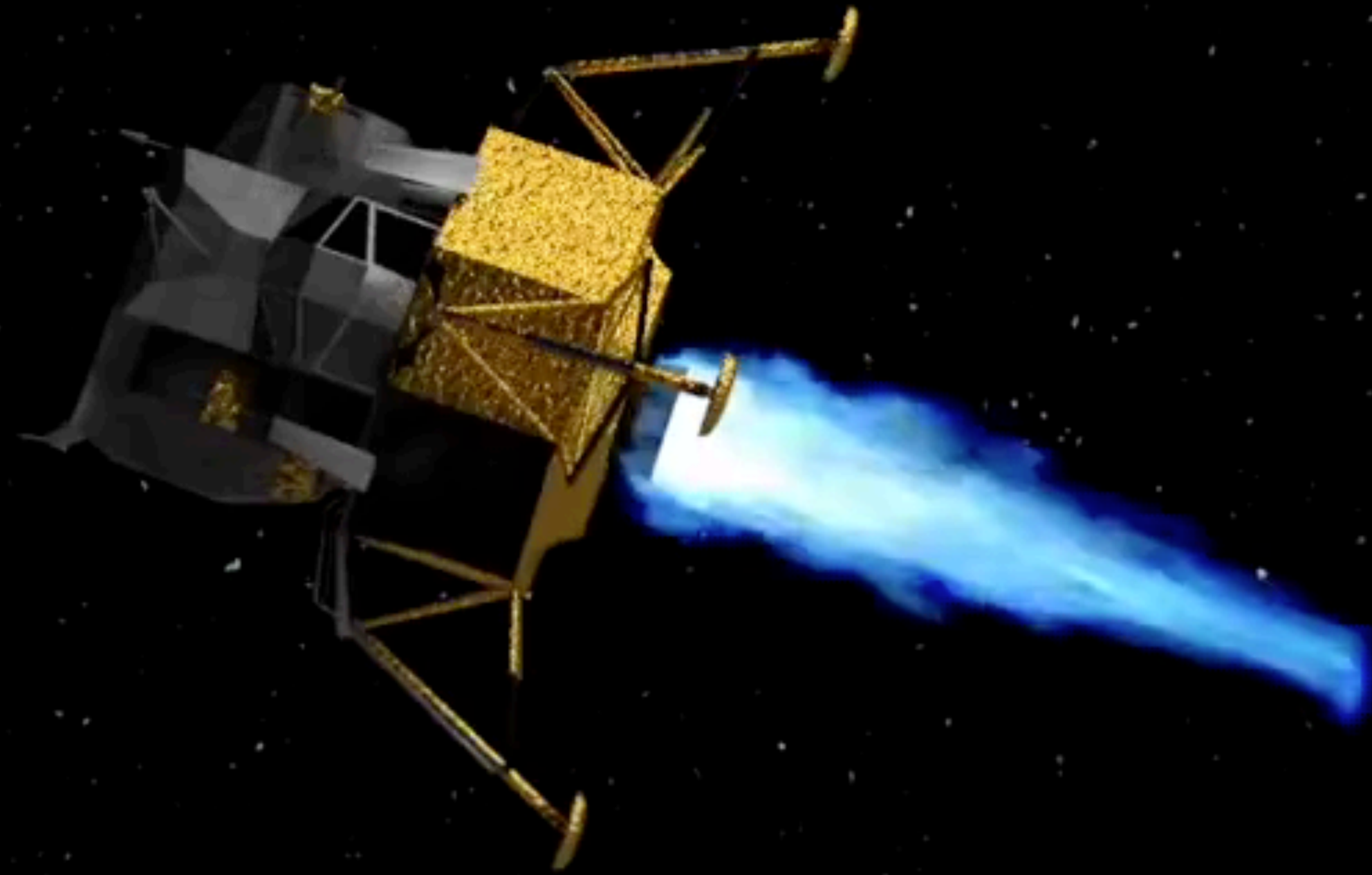


## Powered Descent





## Powered Descent





# Landing





# Landing





# Ascent





Ascent





Ascent





# Lunar Orbit Rendezvous





# Lunar Orbit Rendezvous



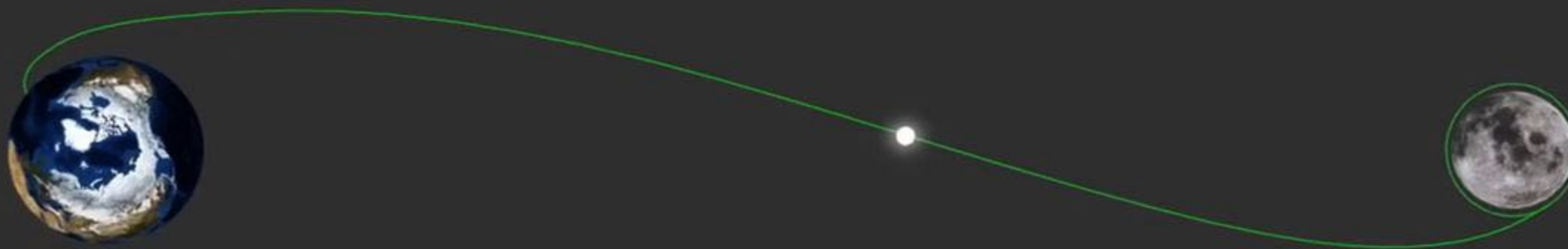


# Lunar Orbit Rendezvous



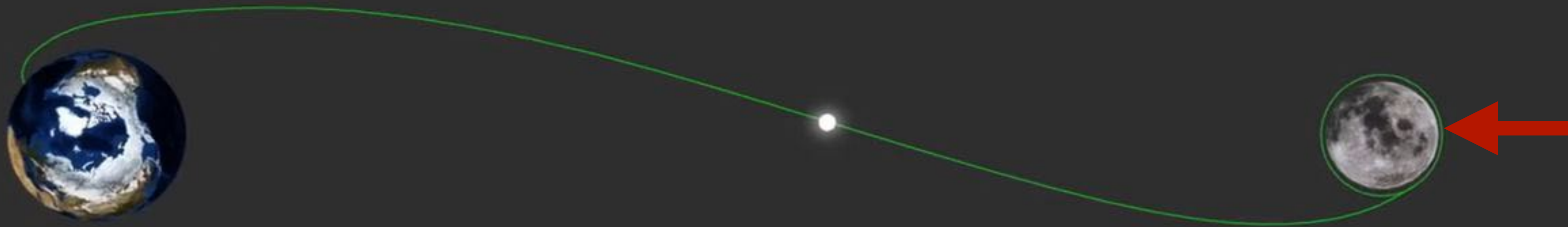


# Trans-Earth Injection



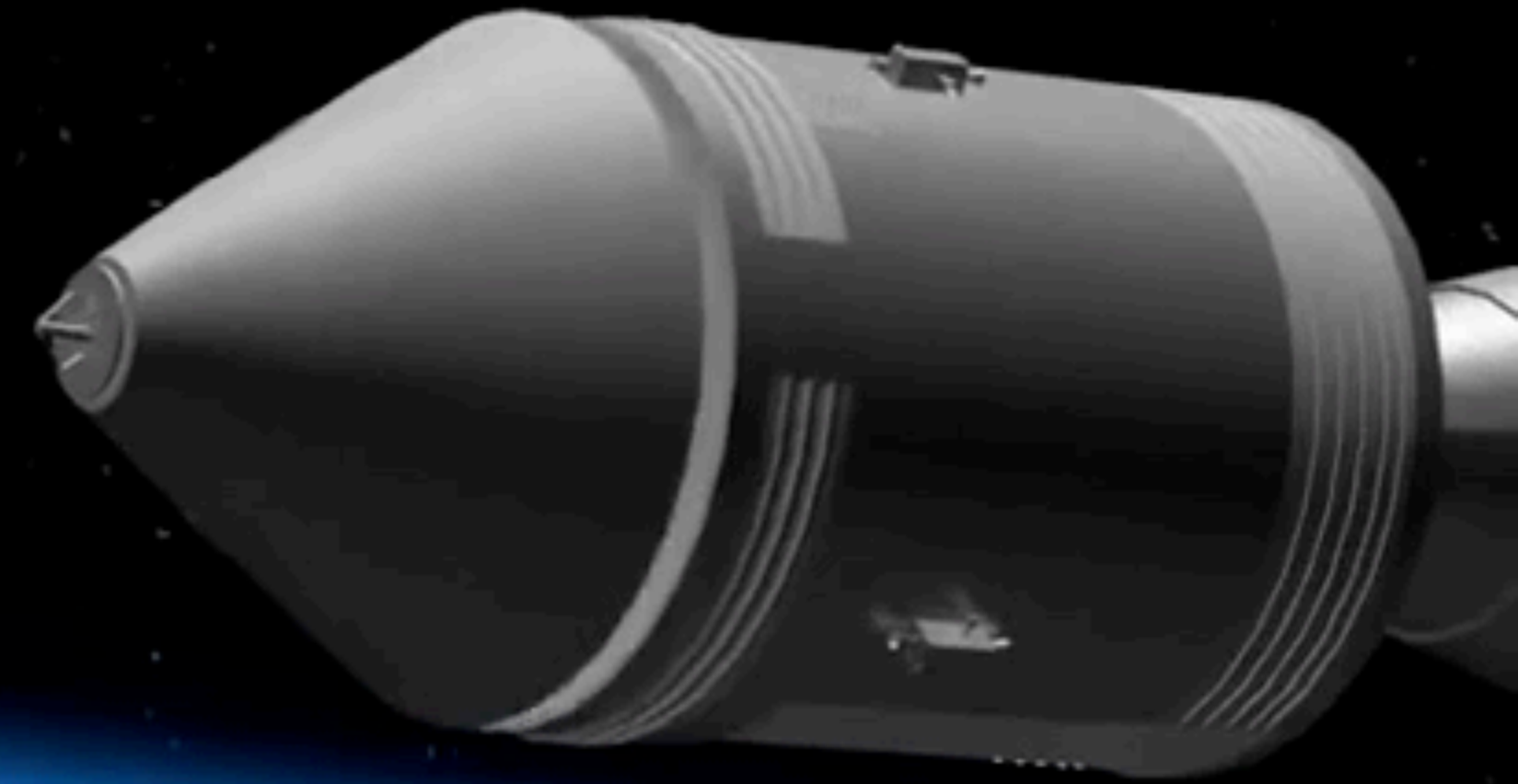


# Trans-Earth Injection



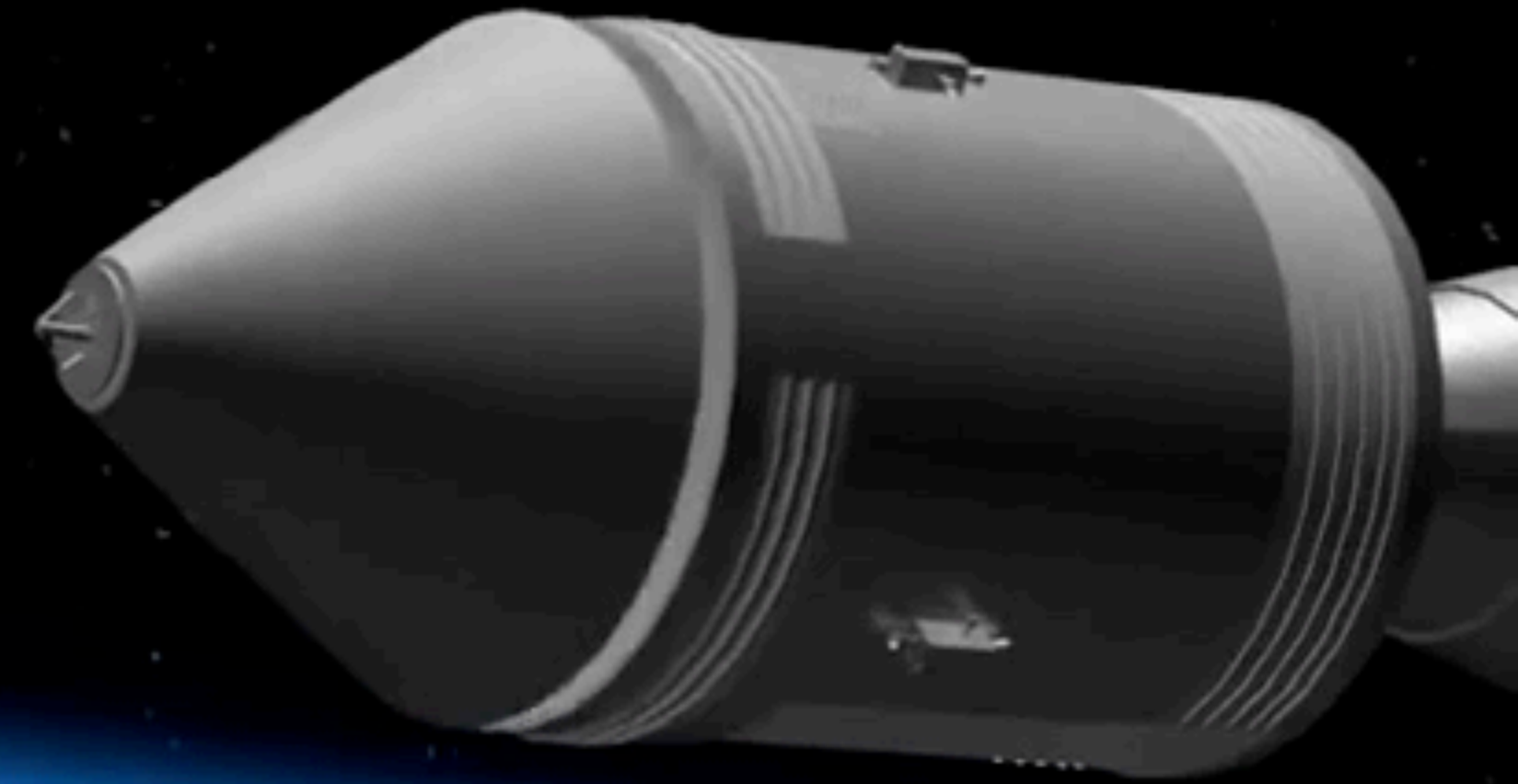


Entry

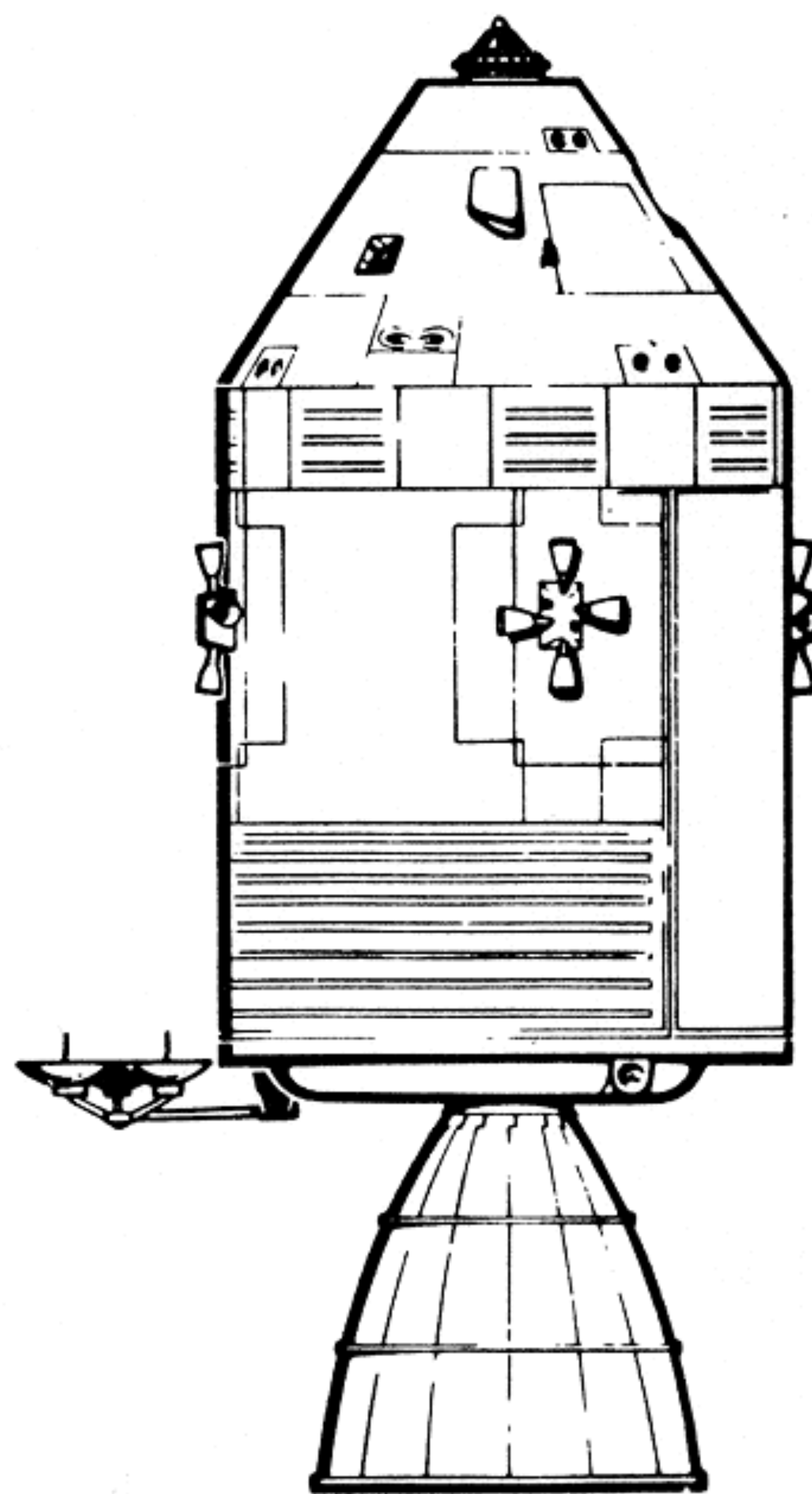




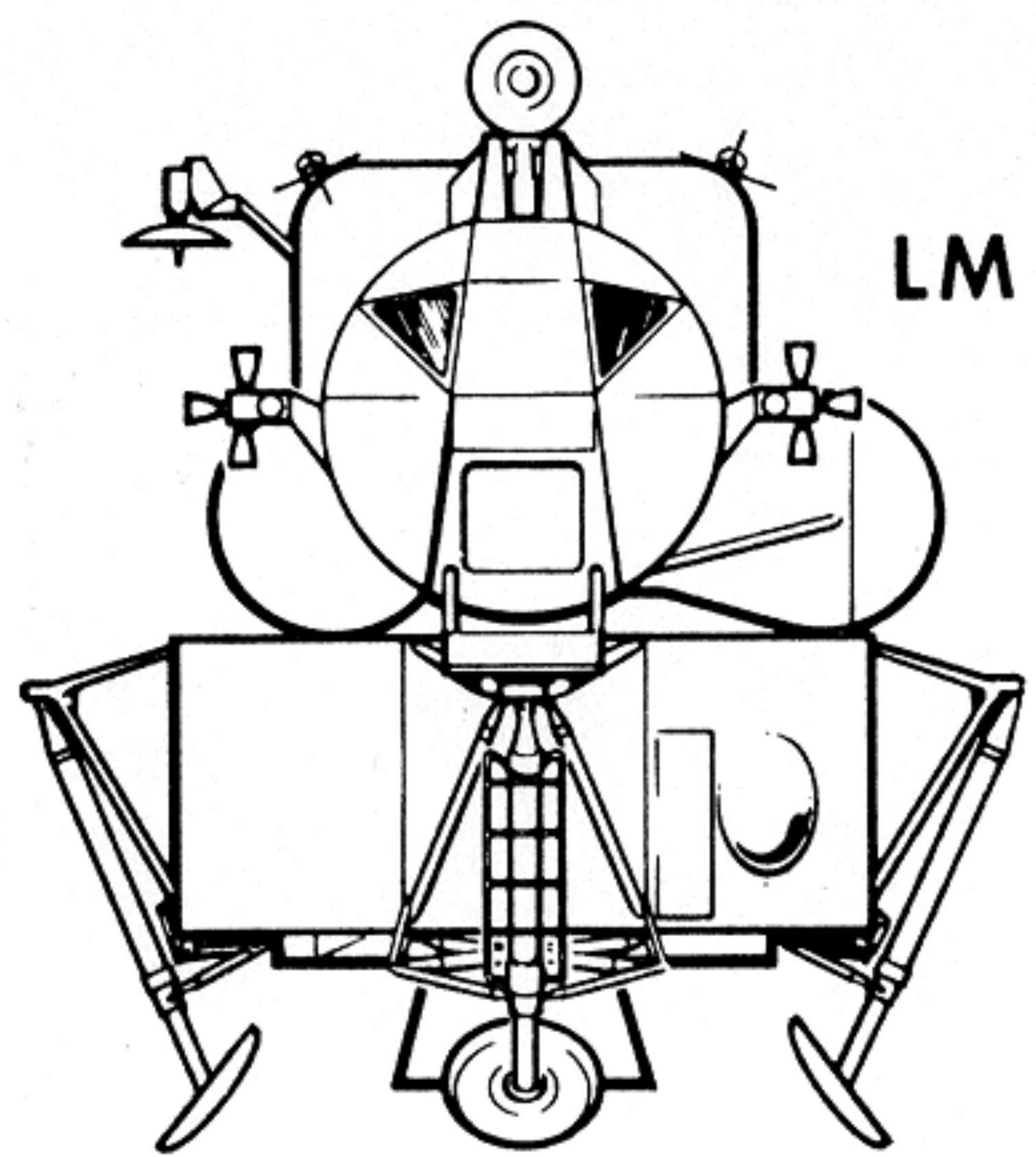
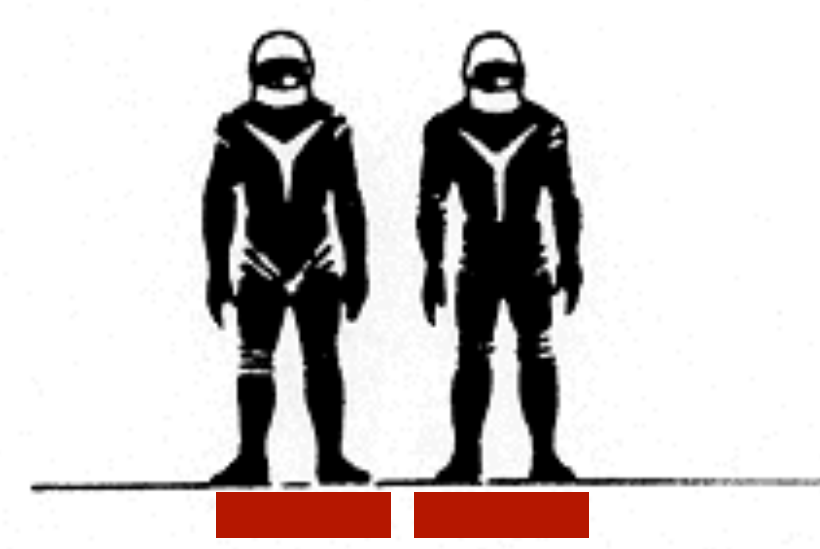
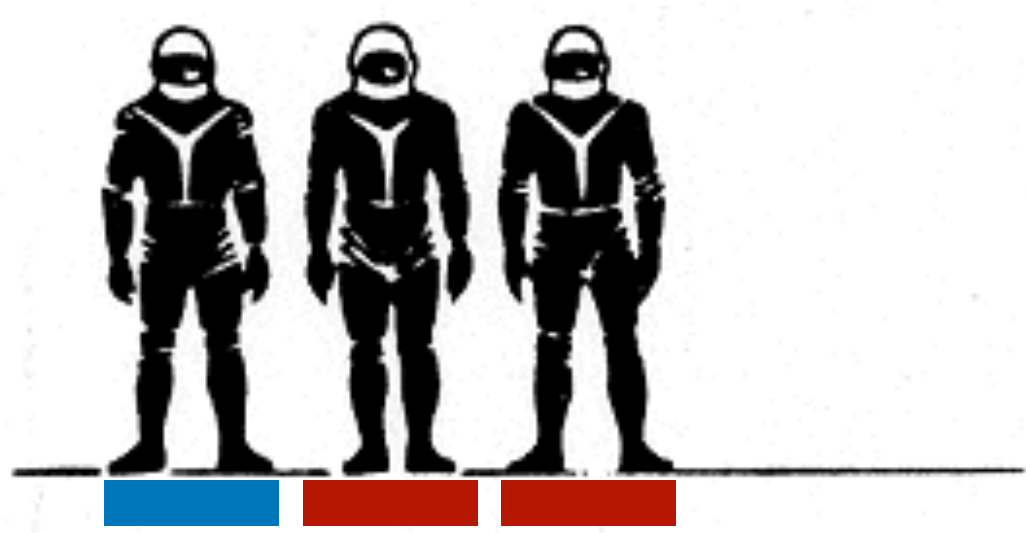
Entry







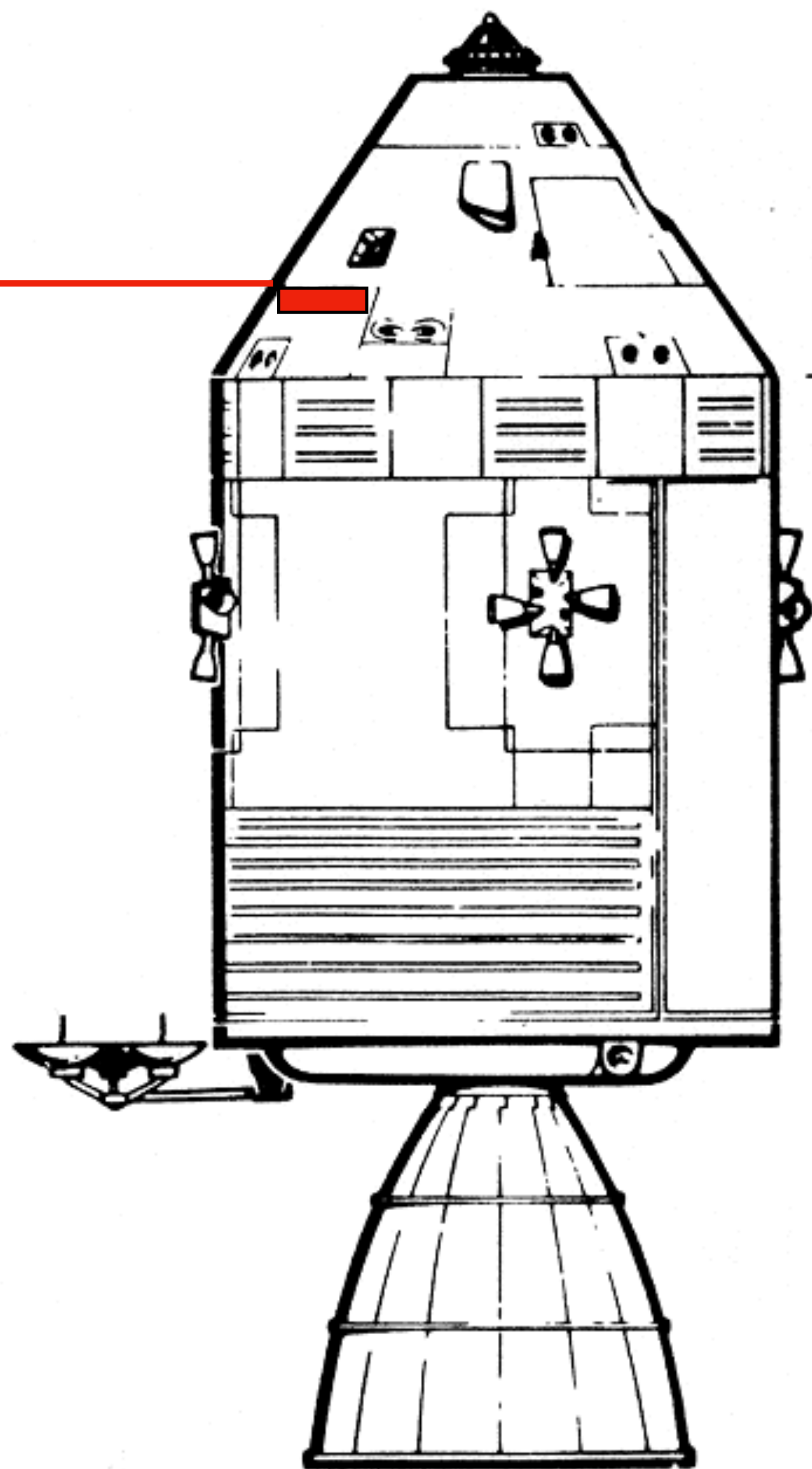
CSM



LM



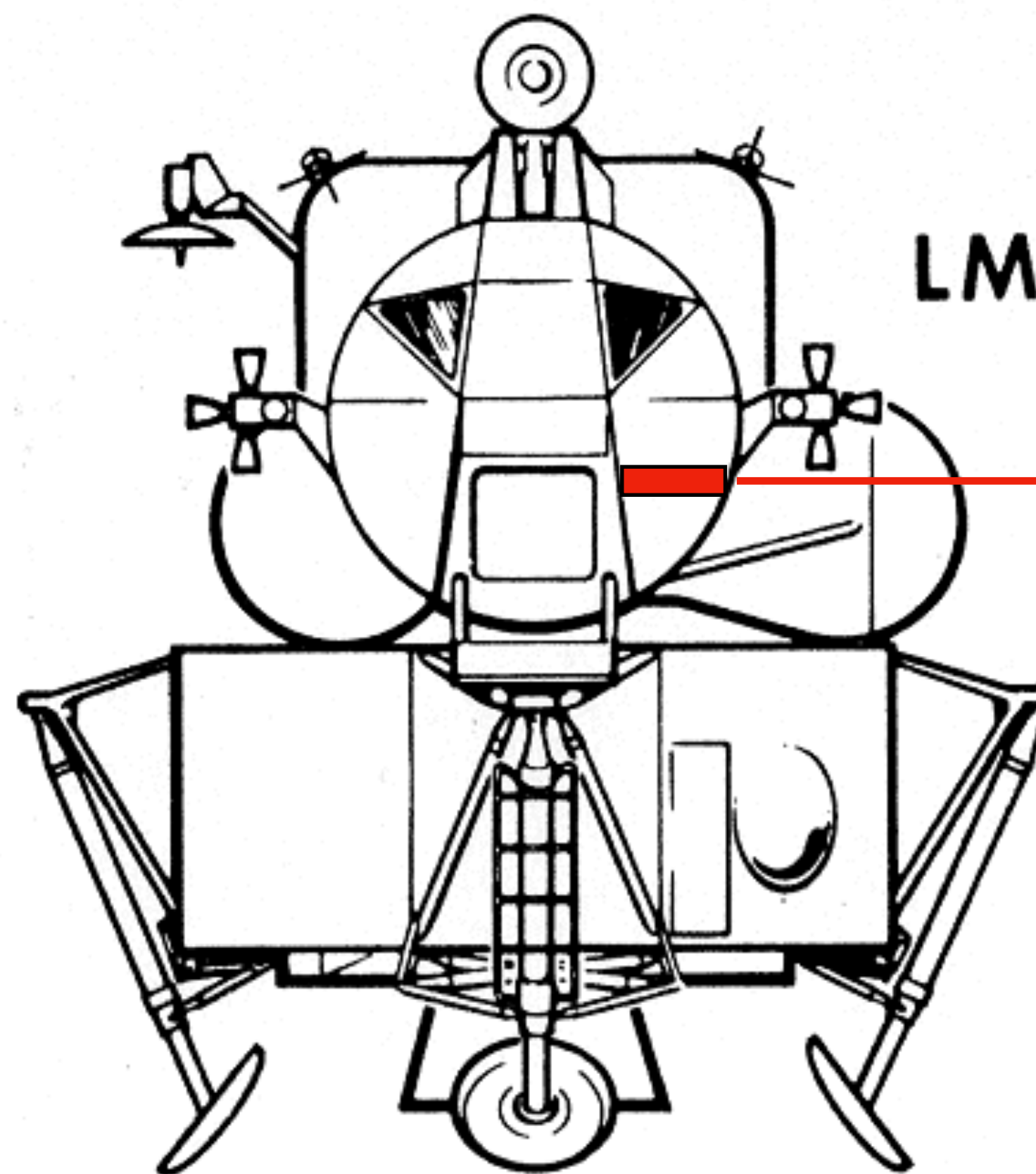
AGC



CSM



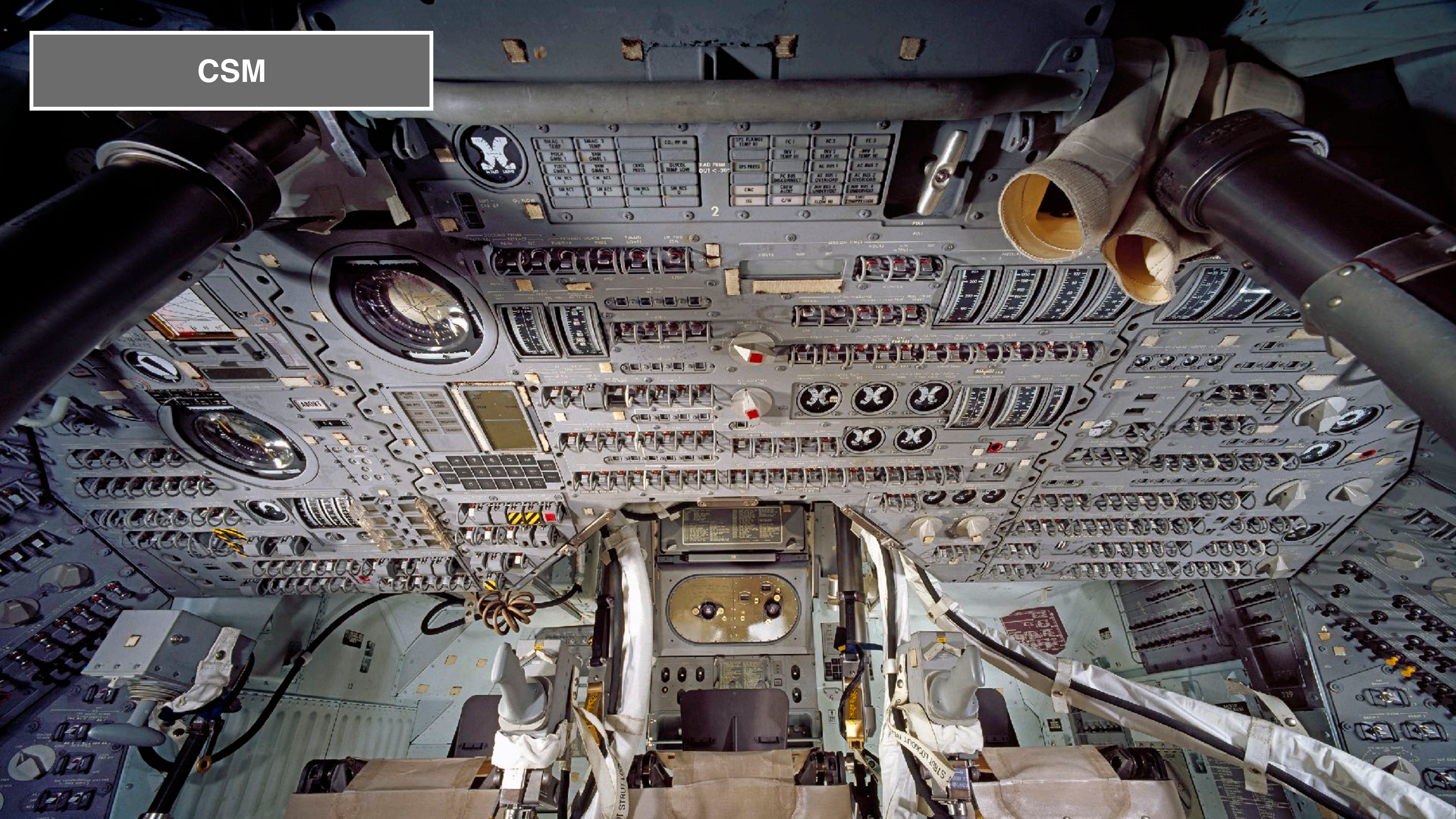
LM



AGC



CSM





CSM





LM

COMMANDER'S CONTROL  
PANEL

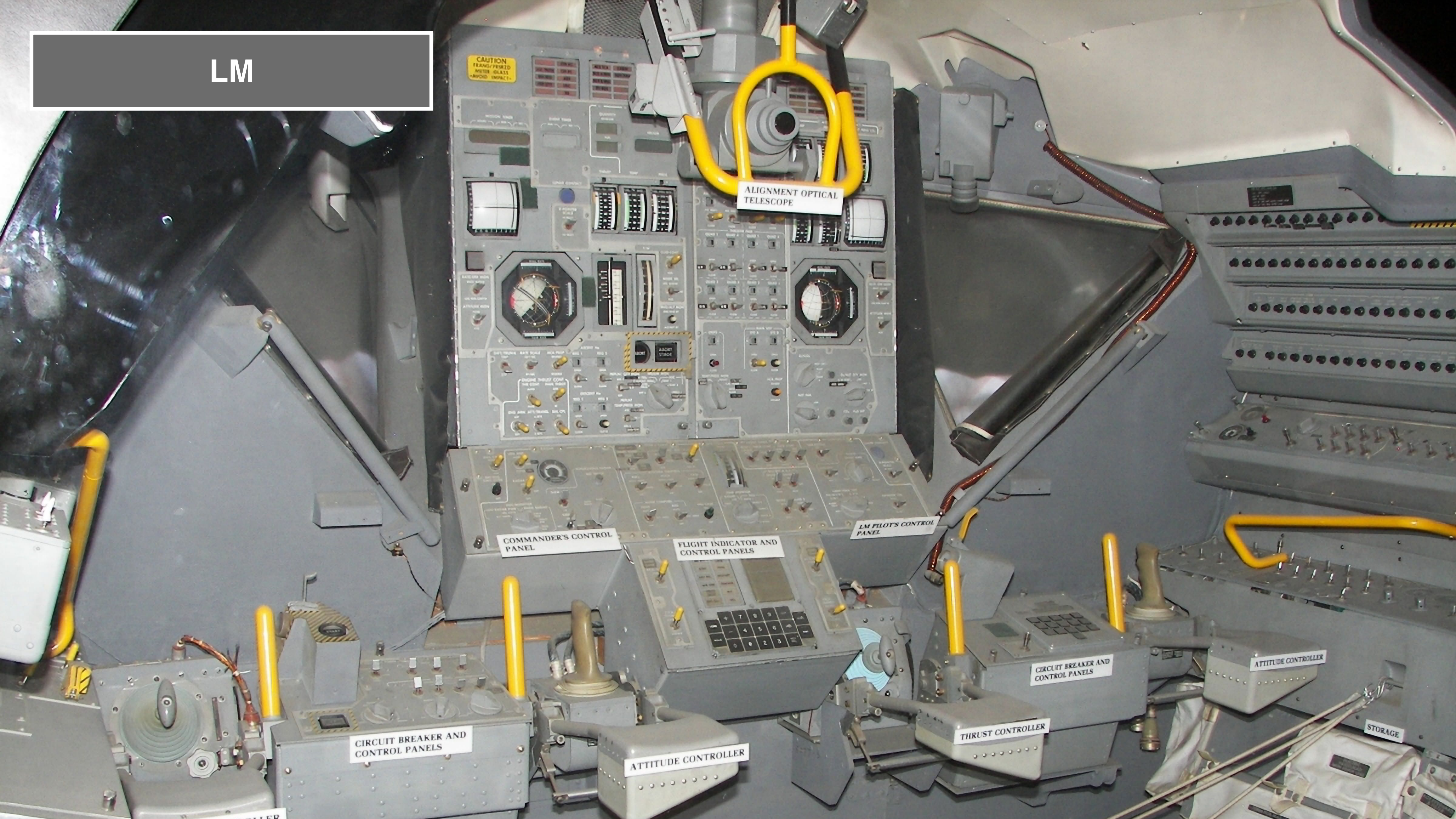
FLIGHT INDICATOR AND  
CONTROL PANELS

LM PILOT'S CONTROL  
PANEL



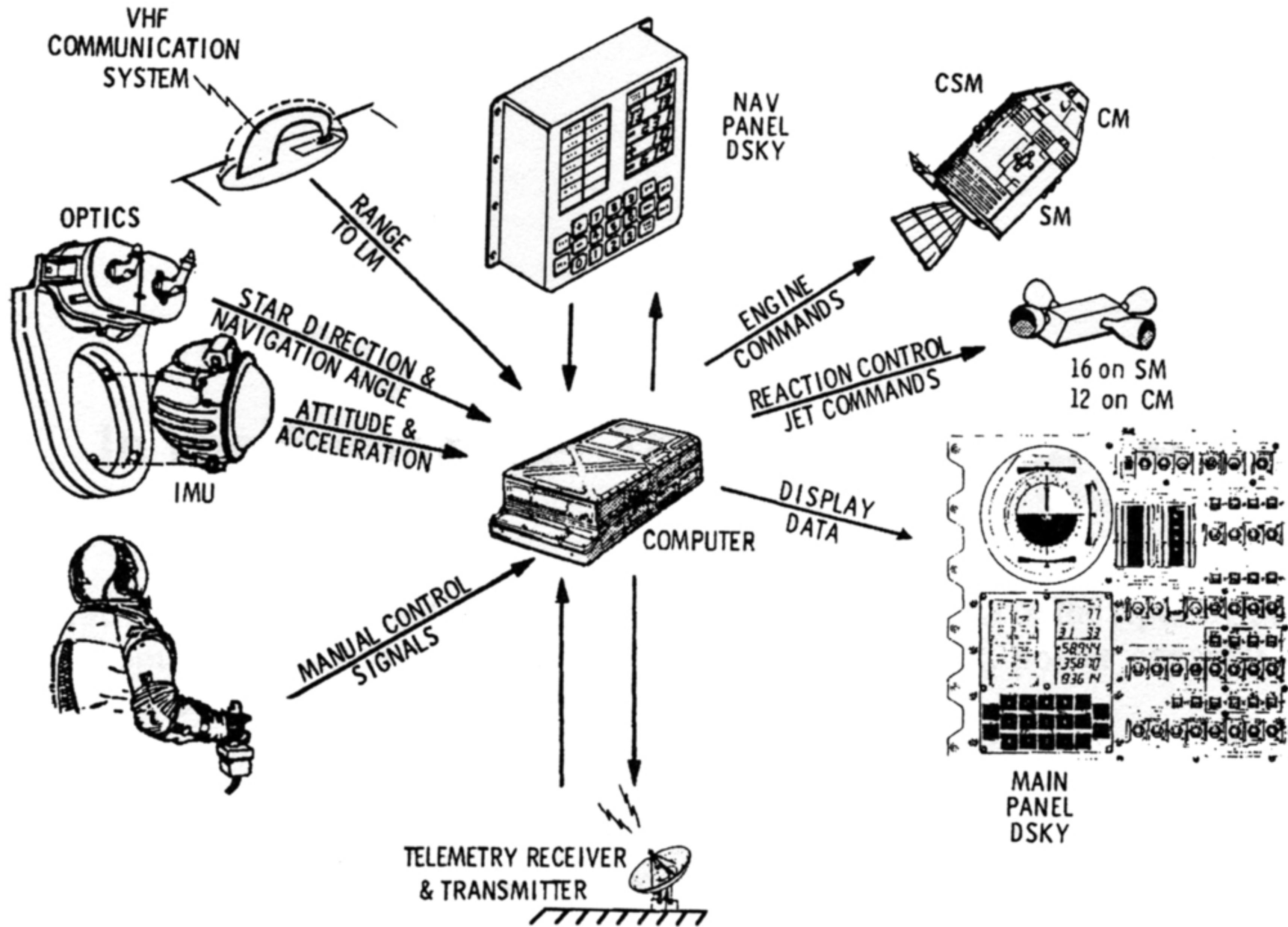


LM



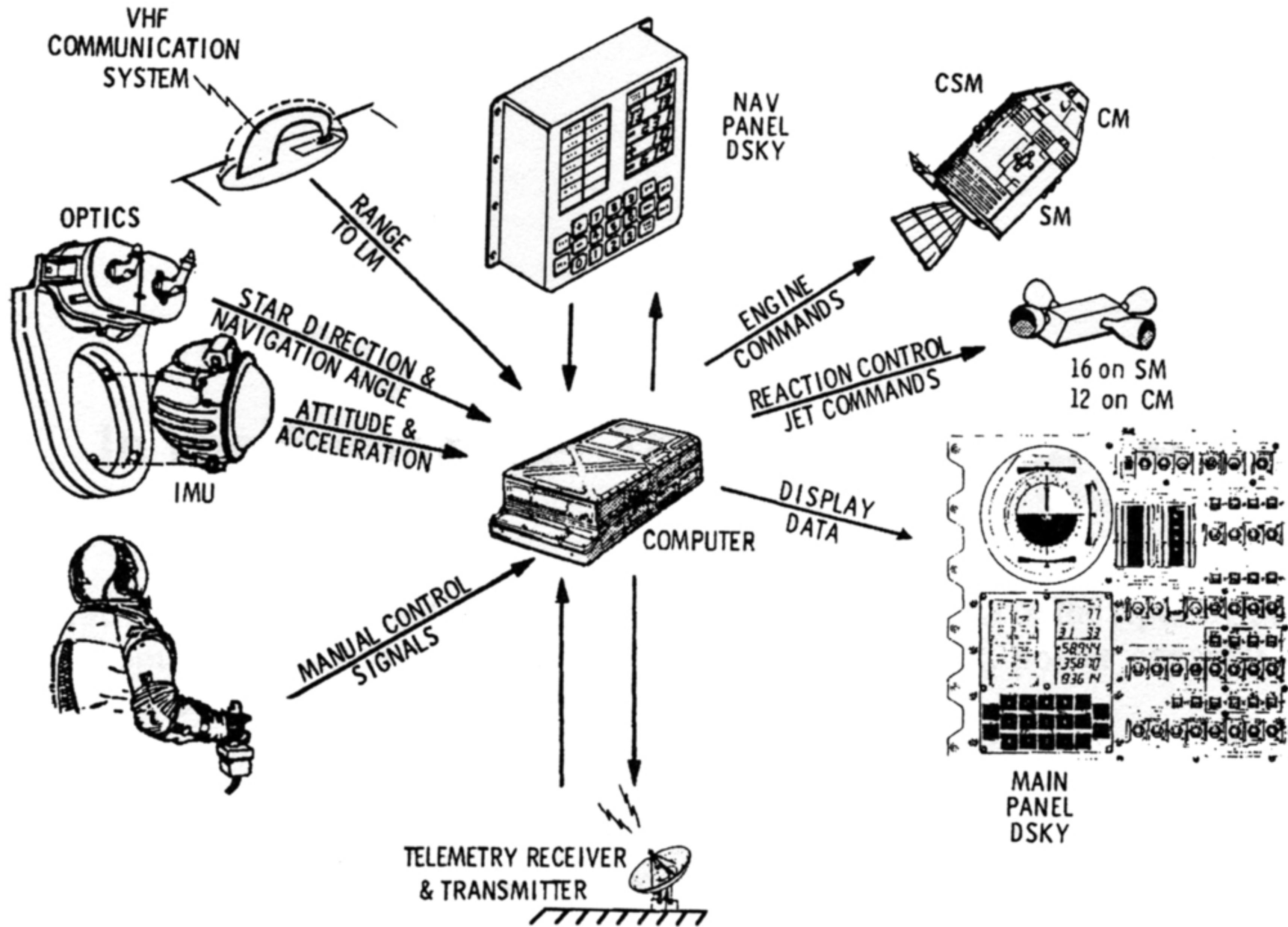


# AGC Responsibilities





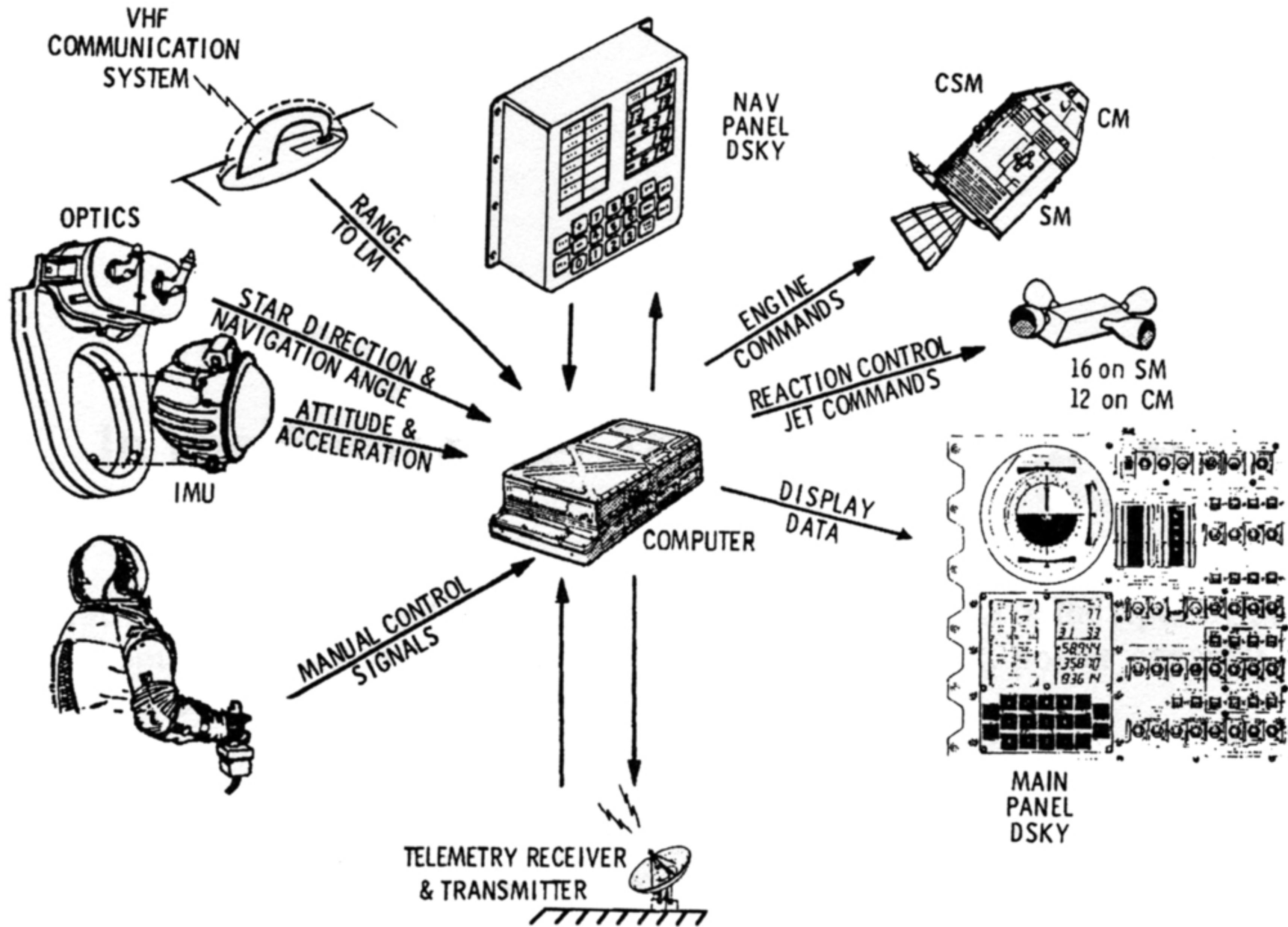
## AGC Responsibilities



## Maintain State Vector



## AGC Responsibilities

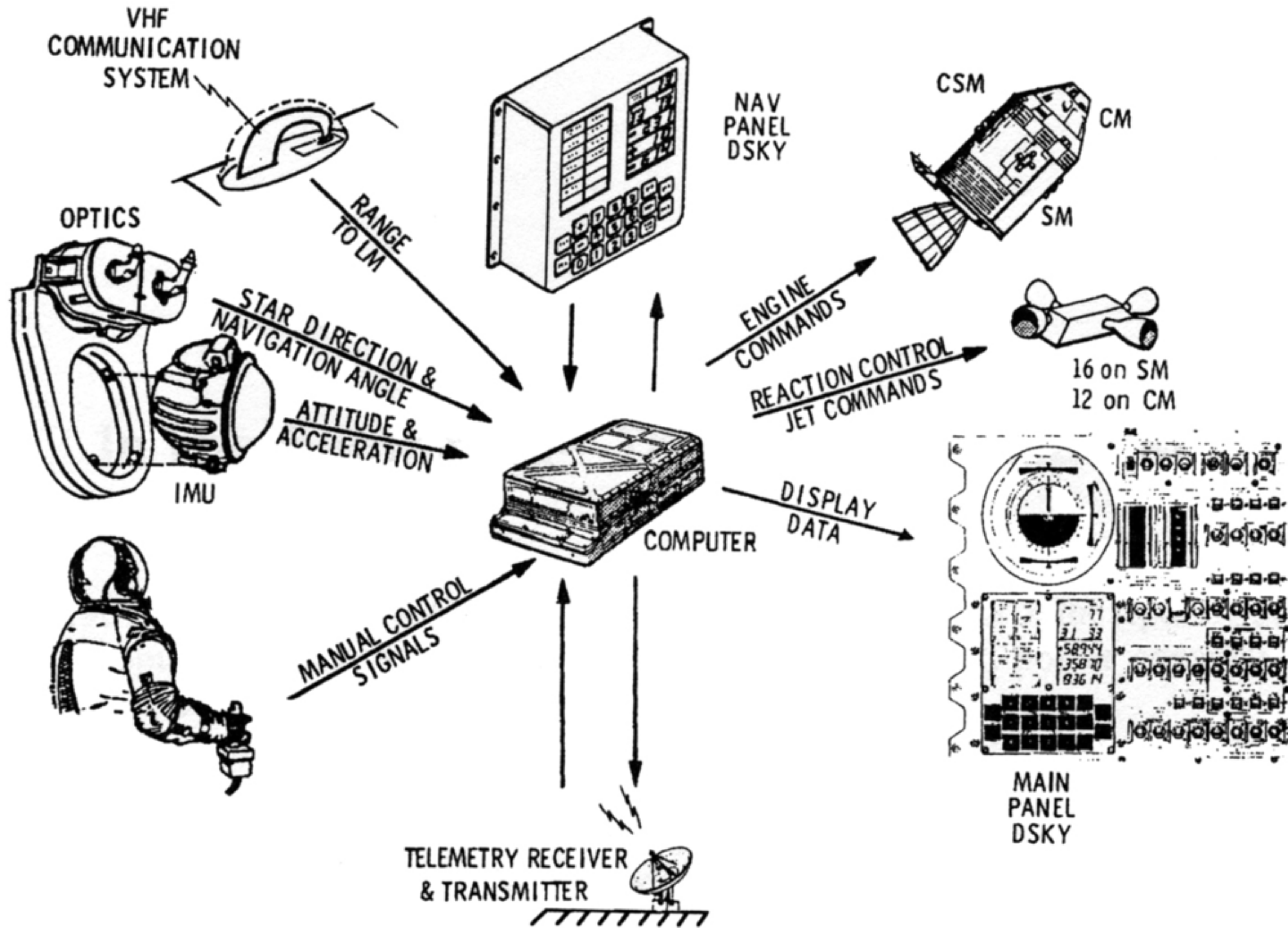


**Maintain State Vector**

**Stabilize Attitude**



## AGC Responsibilities



**Maintain State Vector**

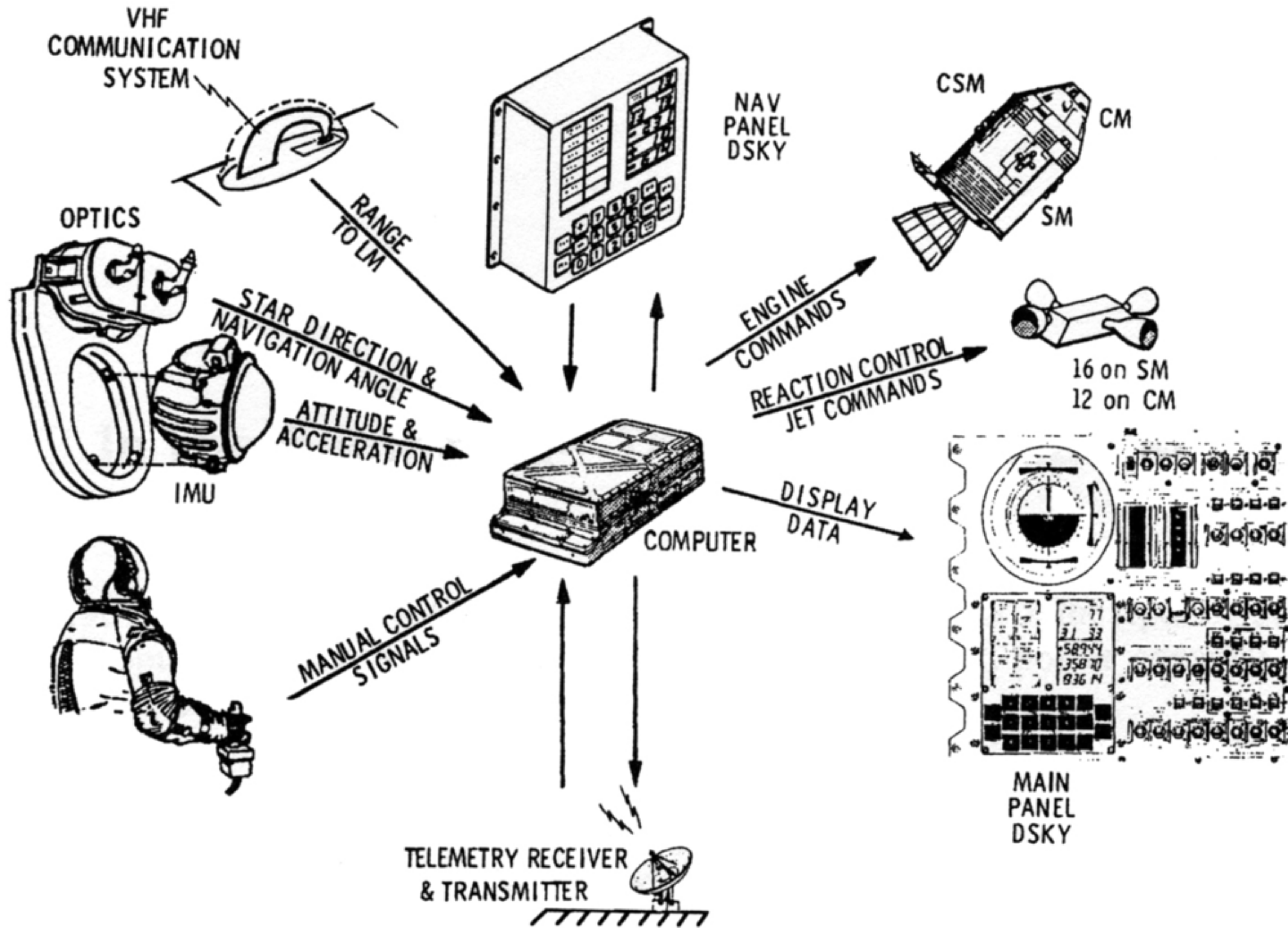
**Stabilize Attitude**

**Calculate Burns**

**Control Burns**



## AGC Responsibilities



**Maintain State Vector**

**Stabilize Attitude**

**Calculate Burns**

**Control Burns**

**Monitor/Control Launch**



**Maintain State  
Vector**

**Calculate Burns**

**Stabilize Attitude**

**Control Burns**

**Monitor/Control  
Launch**



# Apollo Guidance Computer

Architecture

Mission Software

Hardware

Peripherals

System Software



# Apollo Guidance Computer

Architecture

Mission Software

Hardware

Peripherals

System Software



# Apollo Guidance Computer

Architecture

Mission Software

Hardware

Peripherals

System Software



# Apollo Guidance Computer

Architecture

Mission Software

Hardware

Peripherals

System Software



# Apollo Guidance Computer

Architecture

Mission Software

Hardware

Peripherals

System Software



# Apollo Guidance Computer

Architecture

Mission Software

Hardware

Peripherals

System Software



# Apollo Guidance Computer

Architecture

Mission Software

Hardware

Peripherals

System Software



**Architecture**



# Architecture



# Architecture

**von Neumann**

**Accumulator  
Machine**

**15 bit**

**One's  
Complement**

**Big Endian**



# Architecture



**Architecture**

**Instruction Set**

**Arithmetic**

**Encoding**



**Architecture**

**Instruction Set**

**Memory Model**

**Arithmetic**

**I/O**

**Encoding**

**Counters**



**Architecture**

**Instruction Set**

**Memory Model**

**Arithmetic**

**I/O**

**Interrupts**

**Encoding**

**Counters**



# armv8

abs	cmhs	fcvtl2	frinti	ldur	rorv	smaxp	sri	suqadd	umulh
adc	cmle	fcvtms	frintm	ldurb	rshrn	smaxv	srshl	svc	umull
adcs	cmlt	fcvtmu	frintn	ldurh	rshrn2	smc	srshr	sys	umull2
add	cmtst	fcvtn	frintp	ldursb	rsubhn	smin	srsra	sysl	uqadd
addhn	cnt	fcvtn2	frintx	ldursh	rsubhn2	sminp	sshl	tbl	uqrshl
addhn2	csel	fcvtns	frintz	ldursw	saba	sminv	sshr	tbnz	uqrshrn
addp	csinc	fcvtnu	frsqrte	ldxp	sabal	smlal	ssra	tbx	uqrshrn2
adds	csinv	fcvtps	frsqrts	ldxr	sabal2	smlal2	ssubl	tbz	uqshl
addv	csneg	fcvtpu	fsqrt	ldxrb	sabd	smlsl	ssubl2	trn1	uqshrn
adr	dcps1	fcvtxn	fsub	ldxrh	sabdl	smlsl2	ssubw	trn2	uqsub
adrp	dcps2	fcvtxn2	hint	lslv	sabdl2	smov	ssubw2	uaba	uqxtn
aesd	dcps3	fcvtzs	hlt	lsrv	sadalp	smsubl	st1	uabal	uqxtn2
aese	dmb	fcvtzu	hvc	madd	saddl	smulh	st2	uabal2	urecpe
aesimc	drps	fdiv	ins	mla	saddl2	smull	st3	uabd	urhadd
aesmc	dsb	fmadd	isb	mls	saddlp	smull2	st4	uabdl	urshl
and	dup	fmax	ld1	movi	saddlv	sqabs	st1r	uabdl2	urshr
ands	eon	fmaxnm	ld2	movk	saddw	sqadd	st1rb	uadalp	ursqrte
asrv	eor	fmaxnmp	ld3	movn	saddw2	sqdmlal	st1rh	uaddl	ursra
b	eret	fmaxnmv	ld4	movz	sbc	sqdmlal2	st1xp	uaddl2	ushl
b.c	ext	fmaxp	ldar	mrs	sbc	sqdmlsl	st1xr	uaddlp	ushr
bfm	extr	fmaxv	ldarb	msr	sbfm	sqdmlsl2	st1xrb	uaddlv	usqadd
bic	fabd	fmin	ldarh	msub	scvtf	sqdmulh	st1xrh	uaddw	usra
bics	fabs	fminnm	ldaxp	mul	sdiv	sqdmull	stnp	uaddw2	usubl
bif	facge	fminnmp	ldaxr	mvni	shalc	sqdmull2	stp	ubfm	usubl2
bit	facgt	fminnmv	ldaxrb	neg	shalh	sqneg	str	ucvtf	usubw
bl	fadd	fminp	ldaxrh	not	shalm	sqrdbl	strb	udiv	usubw2
blr	faddp	fminv	ldnp	orn	shalp	sqrshl	strh	uhadd	uzp1
br	fccmp	fmla	ldp	orr	shalsu0	sqrshrn	sttr	uhsub	uzp2
brk	fccmpe	fmls	ldpsw	pmul	shalsu1	sqrshrn2	sttrb	umaddl	xtn
bsl	fcmeq	fmov	ldr	pmull	sha256h	sqrshrun	sttrh	umax	xtn2
cbnz	fcnge	fmsub	ldrb	pmull2	sha256h2	sqrshrun2	stur	umaxp	zip1
cbz	fcngt	fmul	ldrh	prfm	sha256su0	sqshl	sturb	umaxv	zip2
ccmn	fcmlt	fmulx	ldrsb	prfum	sha256su1	sqshlu	sturh	umin	
ccmp	fcmlt	fneg	ldrsh	raddhn	shadd	sqshrn	stxp	uminp	
clrex	fcmp	fnmadd	ldrsw	raddhn2	shl	sqshrun	stxr	uminv	
cls	fcmp	fnmsub	ldtr	rbit	shll	sqsub	stxrb	umlal	
clz	fcmlt	fnmul	ldtrb	ret	shll2	sqxtn	stxrh	umlal2	
cmeq	fcvt	frecpe	ldtrh	rev	shsub	sqxtn2	sub	umls1	
cmge	fcvtas	frecps	ldtrsb	rev16	sli	sqxtun	subhn	umls12	
cmgt	fcvtau	frecpx	ldtrsh	rev32	smaddl	sqxtun2	subhn2	umov	
cmhi	fcvtil	frinta	ldtrsw	rev64	smax	srhadd	subs	umsubl	



# subleq

# armv8

subleq a, b, c

abs	cmhs	fcvtl2	frinti	ldur	rorv	smaxp	sri	suqadd	umulh
adc	cmle	fcvtms	frintm	ldurb	rshrn	smaxv	srshl	svc	umull
adcs	cmlt	fcvtmu	frintn	ldurh	rshrn2	smc	srshr	sys	umull2
add	cmtst	fcvtn	frintp	ldursb	rsubhn	smin	srsra	sysl	uqadd
addhn	cnt	fcvtn2	frintx	ldursh	rsubhn2	sminp	ssh1	tbl	uqrsh1
addhn2	cse1	fcvtns	frintz	ldursw	saba	sminv	sshr	tbnz	uqrshrn
addp	csinc	fcvtnu	frsqrte	ldxp	sabal	smla1	ssra	tbx	uqrshrn2
adds	csinv	fcvtps	frsqrts	ldxr	sabal2	smla12	ssubl	tbz	uqsh1
addv	csneg	fcvtpu	fsqrt	ldxrb	sabd	smls1	ssubl2	trn1	uqshrn
adr	dcps1	fcvtxn	fsub	ldxrh	sabd1	smls12	ssubw	trn2	uqsub
adrp	dcps2	fcvtxn2	hint	lslv	sabd12	smov	ssubw2	uaba	uqxtn
aesd	dcps3	fcvtzs	hlt	lsrv	sadalp	smsubl	st1	uabal	uqxtn2
aese	dmb	fcvtzu	hvc	madd	sadd1	smulh	st2	uabal2	urecpe
aesimc	drps	fdiv	ins	m1a	sadd12	smull	st3	uabd	urhadd
aesmc	dsb	fmadd	isb	mls	saddlp	smull2	st4	uabdl	ursh1
and	dup	fmax	ld1	movi	saddlv	sqabs	st1r	uabdl2	urshr
ands	eon	fmaxnm	ld2	movk	saddw	sqadd	st1rb	uadalp	ursqrte
asrv	eor	fmaxnmp	ld3	movn	saddw2	sqdmla1	st1rh	uadd1	ursra
b	eret	fmaxnmv	ld4	movz	sbc	sqdmla12	st1xp	uadd12	ush1
b.c	ext	fmaxp	ldar	mrs	sbc1	sqdmls1	st1xr	uaddlp	ushr
bfm	extr	fmaxv	ldarb	msr	sbfm	sqdmls12	st1xrb	uaddlv	usqadd
bic	fabd	fmin	ldarh	msub	scvtf	sqdmulh	st1xrh	uaddw	usra
bics	fabs	fminnm	ldaxp	mul	sdiv	sqdmull	stnp	uaddw2	usubl
bif	facge	fminnmp	ldaxr	mvni	shalc	sqdmull2	stp	ubfm	usubl2
bit	facgt	fminnmv	ldaxrb	neg	shalh	sqneg	str	ucvtf	usubw
bl	fadd	fminp	ldaxrh	not	shalm	sqrdblh	strb	udiv	usubw2
blr	faddp	fminv	ldnp	orn	shalp	sqrsh1	strh	uhadd	uzp1
br	fccmp	fmla	ldp	orr	shalsu0	sqrshrn	sttr	uhsub	uzp2
brk	fccmpe	fmls	ldpsw	pmul	shalsu1	sqrshrn2	sttrb	umadd1	xtn
bsl	fcmeq	fmov	ldr	pmull	sha256h	sqrshrun	sttrh	umax	xtn2
cbnz	fcmgc	fmsub	ldrb	pmull2	sha256h2	sqrshrun2	stur	umaxp	zip1
cbz	fcmg1	fmul	ldrh	prfm	sha256su0	sqsh1	sturb	umaxv	zip2
ccmn	fcmlc	fmulx	ldrsb	prfum	sha256su1	sqshlu	sturb	umin	
ccmp	fcmlt	fneg	ldrsh	raddhn	shadd	sqshrn	stxp	uminp	
clrex	fcmp	fnmadd	ldrsu	raddhn2	sh1	sqshrun	stxr	uminv	
cls	fcmpc	fnmsub	ldtr	rbit	sh11	sqsub	stxrb	umlal	
clz	fcse1	fnmul	ldtrb	ret	sh112	sqxtn	stxrh	umlal2	
cmeq	fcvt	frecpc	ldtrh	rev	shsub	sqxtn2	sub	umls1	
cmge	fcvtas	frecps	ldtrsb	rev16	sli	sqxtun	subhn	umls12	
cmgt	fcvtau	frecpx	ldtrsh	rev32	smadd1	sqxtun2	subhn2	umov	
cmhi	fcvtl	frinta	ldtrsu	rev64	smax	srhadd	subs	umsub1	



# subleq

**subleq a, b, c**

**$M[b] \leftarrow M[b] - M[a];$   
**if**  $M[b] \leq 0$  **goto** c;**

# armv8

abs	cmhs	fcvtl2	frinti	ldur	rorv	smaxp	sri	suqadd	umulh
adc	cmle	fcvtms	frintm	ldurb	rshrn	smaxv	srshl	svc	umull
adcs	cmlt	fcvtmu	frintn	ldurh	rshrn2	smc	srshr	sys	umull2
add	cmtst	fcvtn	frintp	ldursb	rsubhn	smin	srsra	sysl	uqadd
addhn	cnt	fcvtn2	frintx	ldursh	rsubhn2	sminp	ssh1	tbl	uqrsh1
addhn2	cse1	fcvtns	frintz	ldursw	saba	sminv	sshr	tbnz	uqrshrn
addp	csinc	fcvtnu	frsqrte	ldxp	sabal	smla1	ssra	tbx	uqrshrn2
adds	csinv	fcvtps	frsqrts	ldxr	sabal2	smla12	ssubl	tbz	uqsh1
addv	csneg	fcvtpu	fsqrt	ldxrb	sabd	smls1	ssubl2	trn1	uqshrn
adr	dcps1	fcvtxn	fsub	ldxrh	sabd1	smls12	ssubw	trn2	uqsub
adrp	dcps2	fcvtxn2	hint	lslv	sabd12	smov	ssubw2	uaba	uqxtn
aesd	dcps3	fcvtzs	hlt	lsrv	sadalp	smsubl	st1	uabal	uqxtn2
aese	dmb	fcvtzu	hvc	madd	sadd1	smulh	st2	uabal2	urecpe
aesimc	drps	fdiv	ins	m1a	sadd12	smull	st3	uabd	urhadd
aesmc	dsb	fmadd	isb	mls	saddlp	smull2	st4	uabdl	ursh1
and	dup	fmax	ld1	movi	saddlv	sqabs	st1r	uabdl2	urshr
ands	eon	fmaxnm	ld2	movk	saddw	sqadd	st1rb	uadalp	ursqrte
asrv	eor	fmaxnmp	ld3	movn	saddw2	sqdmla1	st1rh	uadd1	ursra
b	eret	fmaxnmv	ld4	movz	sbc	sqdmla12	st1xp	uadd12	ush1
b.c	ext	fmaxp	ldar	mrs	sbc1	sqdmls1	st1xr	uaddlp	ushr
bfm	extr	fmaxv	ldarb	msr	sbfm	sqdmls12	st1xrb	uaddlv	usqadd
bic	fabd	fmin	ldarh	msub	scvtf	sqdmulh	st1xrh	uaddw	usra
bics	fabs	fminnm	ldaxp	mul	sdiv	sqdmull	stnp	uaddw2	usubl
bif	facge	fminnmp	ldaxr	mvni	sha1c	sqdmull2	stp	ubfm	usubl2
bit	facgt	fminnmv	ldaxrb	neg	sha1h	sqneg	str	ucvtf	usubw
bl	fadd	fminp	ldaxrh	not	shalm	sqrdrmuh	strb	udiv	usubw2
blr	faddp	fminv	ldnp	orn	sha1p	sqrsh1	strh	uhadd	uzp1
br	fccmp	fmla	ldp	orr	shalsu0	sqrshrn	sttr	uhsub	uzp2
brk	fccmpe	fmls	ldpsw	pmul	sha1su1	sqrshrn2	sttrb	umadd1	xtn
bsl	fcmeq	fmov	ldr	pmull	sha256h	sqrshrun	sttrh	umax	xtn2
cbnz	fcmgc	fmsub	ldrb	pmull2	sha256h2	sqrshrun2	stur	umaxp	zip1
cbz	fcmg1	fmul	ldrh	prfm	sha256su0	sqsh1	sturb	umaxv	zip2
ccmn	fcmlc	fmulx	ldrsb	prfum	sha256su1	sqshlu	sturh	umin	
ccmp	fcmlt	fneg	ldrsh	raddhn	shadd	sqshrn	stxp	uminp	
clrex	fcmp	fnmadd	ldrsw	raddhn2	sh1	sqshrun	stxr	uminv	
cls	fcmpc	fnmsub	ldtr	rbit	sh11	sqsub	stxrb	umlal	
clz	fcse1	fnmul	ldtrb	ret	sh112	sqxtn	stxrh	umlal2	
cmeq	fcvt	frecpe	ldtrh	rev	shsub	sqxtn2	sub	umls1	
cmge	fcvtas	frecps	ldtrsb	rev16	sli	sqxtun	subhn	umls12	
cmgt	fcvtau	frecpx	ldtrsh	rev32	smadd1	sqxtun2	subhn2	umov	
cmhi	fcvt1	frinta	ldtrsw	rev64	smax	srhadd	subs	umsub1	



**subleq**

**armv8**





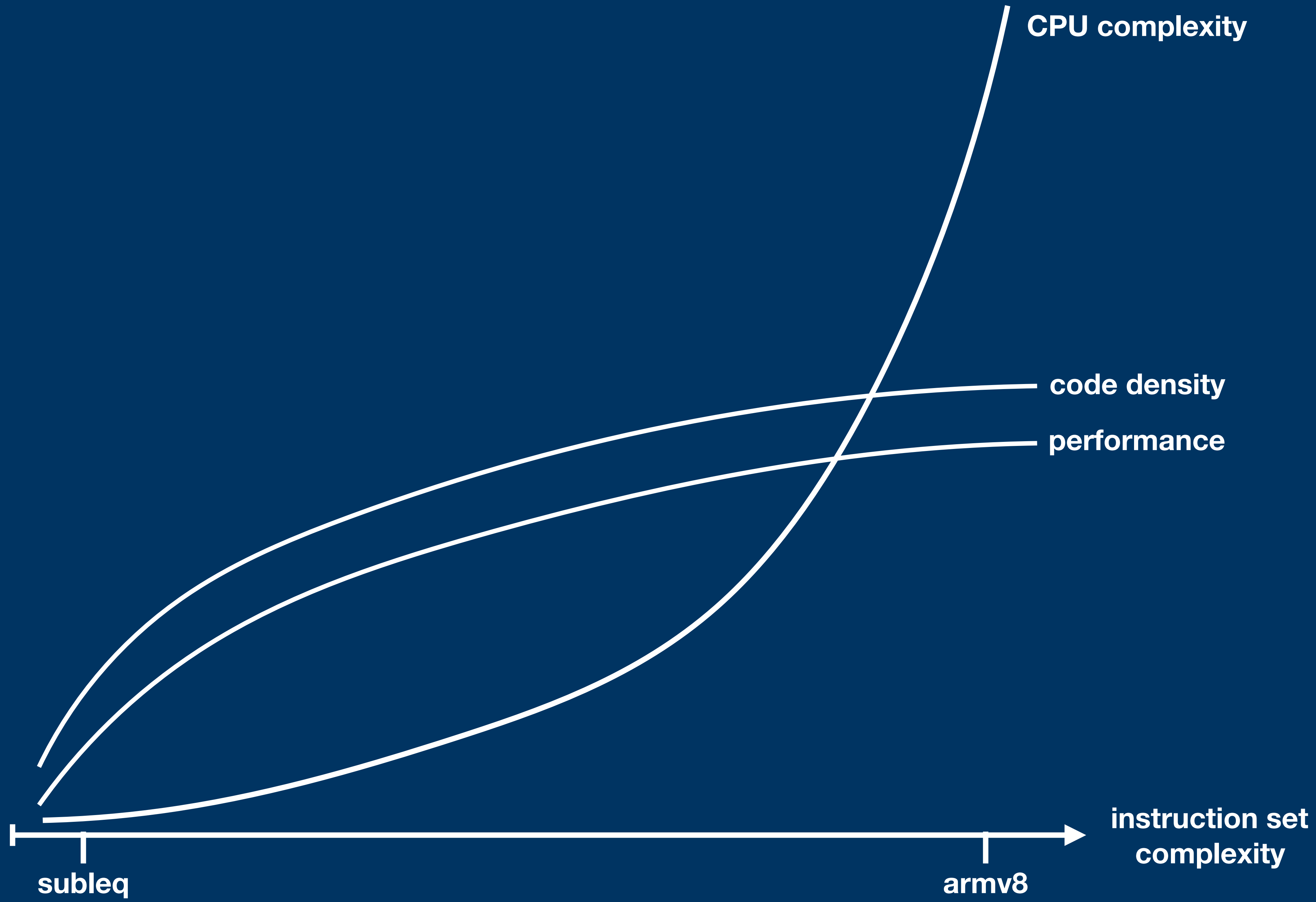




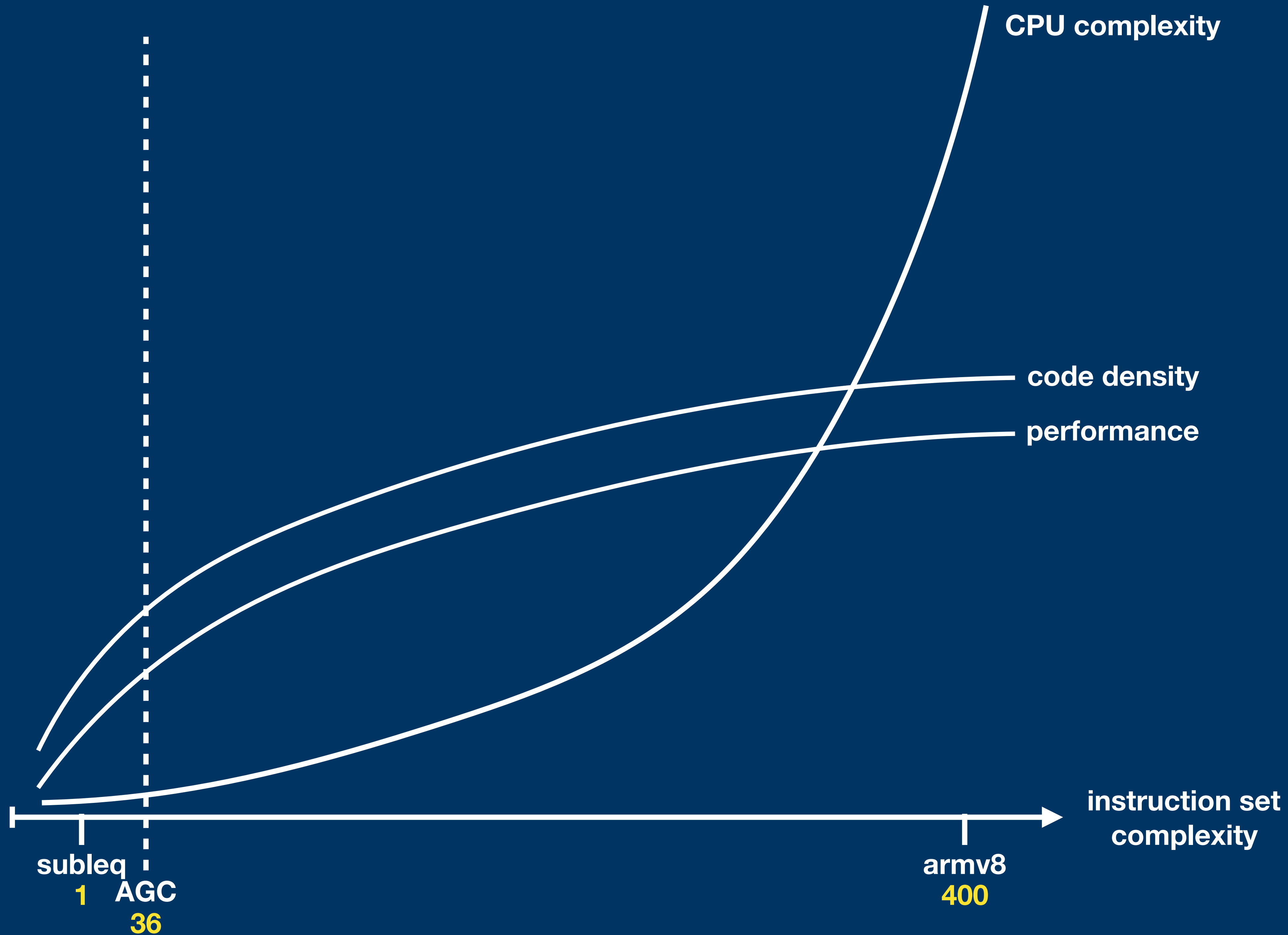


















# Load/Store

Load Cpl	ldc a, [k]	CS k
	4000+k	2
	A ← -M[k]	
Load Dbl Cpl	ldc ab, [k]	DCS k
	0006, 4001+k	4
	A ← -M[k] B ← -M[k+1]	
Load Dbl	ld a, [k]	CA k
	3000+k	2
	A ← M[k]	
xchg	xchg a, [k]	XCH k
	5C00+k	2
	A ↔ M[k]	
Store	ld [k], a	TS k
	5800+k	2
	M[k] ← A	
xchg Dbl	xchg ab, [k]	DXCH k
	5201+k	3
	A ↔ M[k] B ↔ M[k+1]	
xchg B	xchg b, [k]	LXCH k
	2400+k	2
	B ↔ M[k]	
xchg LR	xchg lr, [k]	QXCH k
	0600, 2400+k	3
	LR ↔ M[k]	



# Load/Store

Load Cpl	ldc a, [k]	CS k	Load	ld a, [k]	CA k	Xchg	xchg a, [k]	XCH k	Store	ld [k], a	TS k
	4000+k	2		3000+k	2		5C00+k	2		5800+k	2
	A ← -M[k]			A ← M[k]			A ↔ M[k]			M[k] ← A	
Load Dbl Cpl	ldc ab, [k]	DCS k	Load Dbl	ldc ab, [k]	DC k	Xchg Dbl	xchg ab, [k]	DXCH k			
	0006, 4001+k	4		0006, 3001+k	4		5201+k	3			
	A ← -M[k] B ← -M[k+1]			A ← M[k] B ← M[k+1]			A ↔ M[k] B ↔ M[k+1]				

Add	add a, [k]	AD k	Subtract	sub a, [k]	SU k
	6000+k	2		0006, 6000+k	3
	A ← A + M[k]			A ← A - M[k]	
Add Strg	add [k]	ADS k	2's Sub	sub2 a, [k]	MSU k
	2C00+k	2		0006, 2000+k	3
	A, M[k] ← A + M[k]			A ← A - <sub>2</sub> M[k]	
Add Double	add [k], ab	DAS k	Increment	inc [k]	INCR k
	2001+k	3		2800+k	2
	M[A-1] ← M[k] + 1			M[k] ← M[k] + 1	
Augment	aug [k]	AUG k	Diminish	dim [k]	DIM k
	0006, 2800+k	3		0006, 2C00+k	3
	if M[k] ≥ +0: M[k] ← M[k] + 1 if M[k] ≤ -0: M[k] ← M[k] - 1			if M[k] > +0: M[k] ← M[k] - 1 if M[k] < -0: M[k] ← M[k] + 1	
Multiply	mul [k]	MP k	Divide	div [k]	DV k
	0006, 7000+k	4		0006, 1000+k	7
	A, B ← A × M[k]			A ← A, B ÷ M[k] B ← A, B % M[k]	

Xchg B	xchg b, [k]	LXCH k
	2400+k	2
	B ↔ M[k]	
Xchg LR	xchg lr, [k]	QXCH k
	0600, 2400+k	3
	LR ↔ M[k]	

AND	and a, [k]	MASK k
	700+k	2
	A ← A & M[k]	

# Logic



# Load/Store

Load Cpl	ldc a, [k]	CS k	Load	ld a, [k]	CA k	Xchg	xchg a, [k]	XCH k	Store	ld [k], a	TS k				
	4000+k			2	3000+k		2	5C00+k		2	5800+k		2		
	A ← -M[k]			A ← M[k]			A ↔ M[k]			M[k] ← A					
Load Dbl Cpl	ldc ab, [k]	DCS k	Load Dbl	ldc ab, [k]	DCS k	Xchg Dbl	xchg ab, [k]	DXCH k							
	0006, 4001+k			4	0006, 3001+k		4	5201+k		3					
	A ← -M[k] B ← -M[k+1]			A ← M[k] B ← M[k+1]			A ↔ M[k] B ↔ M[k+1]								

Add	add a, [k]	AD k	Subtract	sub a, [k]	SU k
	6000+k	2		0006, 6000+k	3
	A ← A + M[k]			A ← A - M[k]	
Add Strg	add [k]	ADS k	2's Sub	sub2 a, [k]	MSU k
	2C00+k	2		0006, 2000+k	3
	A, M[k] ← A + M[k]			A ← A - <sub>2</sub> M[k]	
Add Double	add [k], ab	DAS k	Increment	inc [k]	INCR k
	2001+k	3		2800+k	2
	M[A-1] ← M[k] + 1			M[k] ← M[k] + 1	
Augment	aug [k]	AUG k	Diminish	dim [k]	DIM k
	0006, 2800+k	3		0006, 2C00+k	3
	if M[k] ≥ +0: M[k] ← M[k] + 1 if M[k] ≤ -0: M[k] ← M[k] - 1			if M[k] > +0: M[k] ← M[k] - 1 if M[k] < -0: M[k] ← M[k] + 1	
Multiply	mul [k]	MP k	Divide	div [k]	DV k
	0006, 7000+k	4		0006, 1000+k	7
	A, B ← A × M[k]			A ← A, B ÷ M[k] B ← A, B % M[k]	

Xchg B	xchg b, [k]	LXCH k
	2400+k	2
	B ↔ M[k]	
Xchg LR	xchg lr, [k]	QXCH k
	0600, 2400+k	3
	LR ↔ M[k]	

AND	and a, [k]	MASK k
	700+k	2
	A ← A & M[k]	

# Control Flow

Jump	jmp k	TCF k
	1000+k	1
	PC ← k	
Jump = 0	jz k	BZF k
	0006, 1000+k	2-3
	if A=±0: PC ← k	
Jump ≤ 0	jlez k	BZMF k
	0006, 6000+k	2-3
	if A ≤ ±0: PC ← k	
Count, Compare and Skip	cc [k]	CC k
	1000+k	2
	PC ← DIM(M[k]) if M[k] > 0: PC ← PC + 1 if M[k] < -0: PC ← PC + 2 if M[k] = -0: PC ← PC + 3	
Call	call k	TC k
	0000+k	1
	LR ← PC PC ← k	



# Load/Store

Load Cpl	ldc a, [k]	CS k
	4000+k	2
	$A \leftarrow -M[k]$	
Load Dbl Cpl	ldc ab, [k]	DCS k
	0006, 4001+k	4
	$A \leftarrow -M[k]$ $B \leftarrow -M[k+1]$	

Load	ld a, [k]	CA k
	3000+k	2
	$A \leftarrow M[k]$	
Load Doub	ld ab, [k]	CA k
	0006, 3001+k	4
	$A \leftarrow M[k]$ $B \leftarrow M[k+1]$	

Xchg	xchg a, [k]	XCH k
	5C00+k	2
	$A \leftrightarrow M[k]$	
Xchg Dbl	xchg ab, [k]	DXCH k
	5201+k	3
	$A \leftrightarrow M[k]$ $B \leftrightarrow M[k+1]$	

Store	ld [k], a	TS k
	5800+k	2
	$M[k] \leftarrow A$	

Xchg B	xchg b, [k]	LXCH k
	2400+k	2
	$B \leftrightarrow M[k]$	
Xchg LR	xchg lr, [k]	QXCH k
	0600, 2400+k	3
	$LR \leftrightarrow M[k]$	

Jump	jmp k	TCF k
	1000+k	1
	$PC \leftarrow k$	
Jump = 0	jz k	BZF k
	0006, 1000+k	2-3
	if A=±0: $PC \leftarrow k$	
Jump ≤ 0	jlez k	BZMF k
	0006, 6000+k	2-3
	if A≤±0: $PC \leftarrow k$	
Count, Compare and Skip	cc [k]	CC k
	1000+k	2
	$CC \leftarrow DIM(M[k])$ if $M[k] > 0$ : $PC \leftarrow PC + 1$ if $M[k] < -0$ : $PC \leftarrow PC + 2$ if $M[k] = -0$ : $PC \leftarrow PC + 3$	
Call	call k	TC k
	0000+k	1
	$LR \leftarrow PC$ $PC \leftarrow k$	

In	in a, [kc]	READ kc
	0006, 0000+kc	3
	$A \leftarrow IO[kc]$	
In AND	in& a, [kc]	RAND kc
	0006, 0400+kc	3
	$A \leftarrow A \& IO[kc]$	
In OR	in  a, [kc]	ROR kc
	0006, 0800+kc	3
	$A \leftarrow A   IO[kc]$	
In XOR	in^ a, [kc]	RXOR k
	0006, 0C00+kc	3
	$A \leftarrow A \wedge IO[kc]$	

Out	out [kc], a	WRITE kc
	0006, 0200+kc	3
	$IO[kc] \leftarrow A$	
Out AND	out& [kc], a	WAND kc
	0006, 0600+kc	3
	$A \wedge IO[kc] \leftarrow A \& IO[kc]$	
Out OR	out  [kc], a	WOR kc
	0006, 0A00+kc	3
	$A, IO[kc] \leftarrow A   IO[kc]$	

# Arithmetic

Add	add a, [k]	AD k
	6000+k	2
	$A \leftarrow A + M[k]$	
Add Strg	add [k]	ADS k
	2C00+k	2
	$A, M[k] \leftarrow A + M[k]$	
Add Double	add [k], ab	DAS k
	2001+k	3
	$M[A-1] \leftarrow M[k] + M[B]$	
Augment	aug [k]	AUG k
	0006, 2800+k	3
	if $M[k] \geq +0$ : $M[k] \leftarrow M[k] + 1$ if $M[k] \leq -0$ : $M[k] \leftarrow M[k] - 1$	
Multiply	mul [k]	MP k
	0006, 7000+k	4
	$A, B \leftarrow A \times M[k]$	

Subtract	sub a, [k]	SU k
	0006, 6000+k	3
	$A \leftarrow A - M[k]$	
2's Sub	sub2 a, [k]	MSU k
	0006, 2000+k	3
	$A \leftarrow A -_2 M[k]$	
Increment	inc [k]	INCR k
	2800+k	2
	$M[k] \leftarrow M[k] + 1$	
Diminish	dim [k]	DIM k
	0006, 2C00+k	3
	if $M[k] > +0$ : $M[k] \leftarrow M[k] - 1$ if $M[k] < -0$ : $M[k] \leftarrow M[k] + 1$	
Divide	div [k]	DV k
	0006, 1000+k	7
	$A \leftarrow A, B \div M[k]$ $B \leftarrow A, B \% M[k]$	

# Logic

AND	and a, [k]	MASK k
	7000+k	2
	$A \leftarrow A \& M[k]$	

# Control Flow



# Load/Store

Load Cpl	ldc a, [k]	CS k	Load	ld a, [k]	CA k	Xchg	xchg a, [k]	XCH k			
	4000+k			2	3000+k		2	5C00+k		2	
	A ← -M[k]			A ← M[k]			A ↔ M[k]				
Load Dbl Cpl	ldc ab, [k]	DCS k	Load Dbl	ldc ab, [k]	DC k	Xchg Dbl	xchg ab, [k]	DXCH k			
	0006, 4001+k			4	0006, 3001+k		4	5201+k		3	
	A ← -M[k] B ← -M[k+1]			A ← M[k] B ← M[k+1]			A ↔ M[k] B ↔ M[k+1]				

Store	ld [k], a	TS k
	5800+k	2
	M[k] ← A	

In	in a, [kc]	READ kc	Out	out [kc], a	WRITE kc
	0006, 0000+kc	3		0006, 0200+kc	3
	A ← IO[kc]			IO[kc] ← A	
In AND	in& a, [kc]	RAND kc	Out AND	out& [kc], a	WAND kc
	0006, 0400+kc	3		0006, 0600+kc	3
	A ← A & IO[kc]			A & IO[kc] ← A & IO[kc]	
In OR	in  a, [kc]	ROR kc	Out OR	out  [kc], a	WOR kc
	0006, 0800+kc	3		0006, 0A00+kc	3
	A ← A   IO[ck]			A, IO[kc] ← A   IO[kc]	
In XOR	in^ a, [kc]	RXOR k	Out XOR	out^ [kc], a	WXOR k
	0006, 0C00+kc	3		0006, 0E00+kc	3
	A ← A ^ IO[kc]			A ^ IO[kc] ← A ^ IO[kc]	

Add	add a, [k]	AD k	Subtract	sub a, [k]	SU k
	6000+k	2		0006, 6000+k	3
	A ← A + M[k]			A ← A - M[k]	
Add Strg	add [k]	ADS k	2's Sub	sub2 a, [k]	MSU k
	2C00+k	2		0006, 2000+k	3
	A, M[k] ← A + M[k]			A ← A - <sub>2</sub> M[k]	
Add Double	add [k], ab	DAS k	Increment	inc [k]	INCR k
	2001+k	3		2800+k	2
	M[k] ← M[k] + 1			M[k] ← M[k] + 1	
Augment	aug [k]	AUG k	Diminish	dim [k]	DIM k
	0006, 2800+k	3		0006, 2C00+k	3
	if M[k] ≥ +0: M[k] ← M[k] + 1 if M[k] ≤ -0: M[k] ← M[k] - 1			if M[k] > +0: M[k] ← M[k] - 1 if M[k] < -0: M[k] ← M[k] + 1	
Multiply	mul [k]	MP k	Divide	div [k]	DV k
	0006, 7000+k	4		0006, 1000+k	7
	A, B ← A × M[k]			A ← A, B ÷ M[k] B ← A, B % M[k]	

AND	and a, [k]	MASK k
	7000+k	2
	A ← A & M[k]	

# Control Flow

Jump	jmp k	TCF k
	1000+k	1
	PC ← k	
Jump = 0	jz k	BZF k
	0006, 1000+k	2-3
	if A=±0: PC ← k	
Jump ≤ 0	jlez k	BZMF k
	0006, 6000+k	2-3
	if A ≤ ±0: PC ← k	
Count, Compare and Skip	cc [k]	CC k
	1000+k	2
	PC ← DIM(M[k]) if M[k] > 0: PC ← PC + 1 if M[k] < -0: PC ← PC + 2 if M[k] = -0: PC ← PC + 3	
Call	call k	TC k
	0000+k	1
	LR ← PC PC ← k	


Interrupt	int k	EDRUPT k
	0006, 0E00+k	4
	(disable int) PC' ← PC IR ← IIR'	
Int On	sti	RELINT
	0003	1
	(enable int)	
Int Return	iret	RESUME
	0017	2
	(enable int) PC ← PC' IIR ← IIR'	
Int Off	cli	INHINT
	0004	1
	(disable int)	

# Interrupts



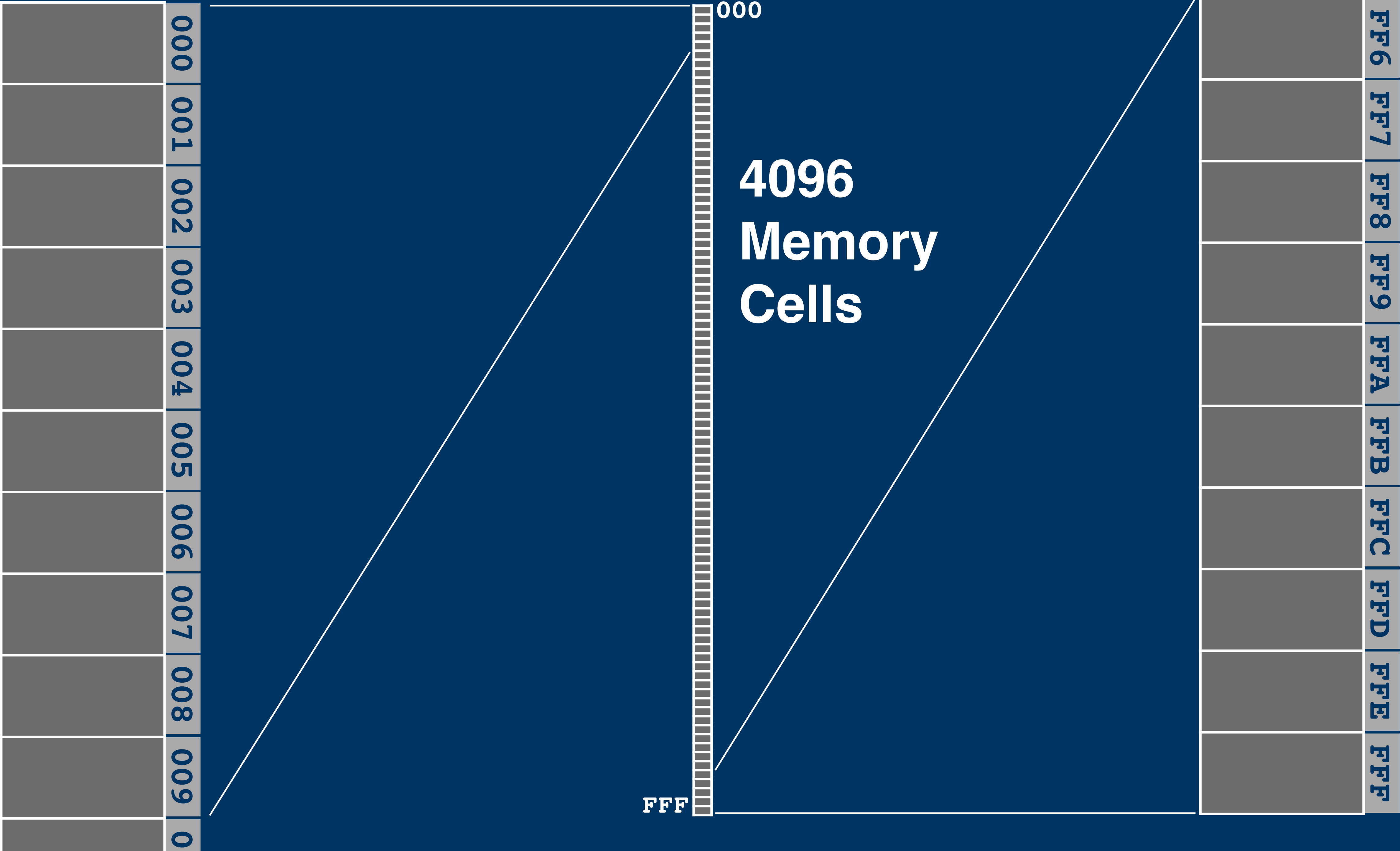




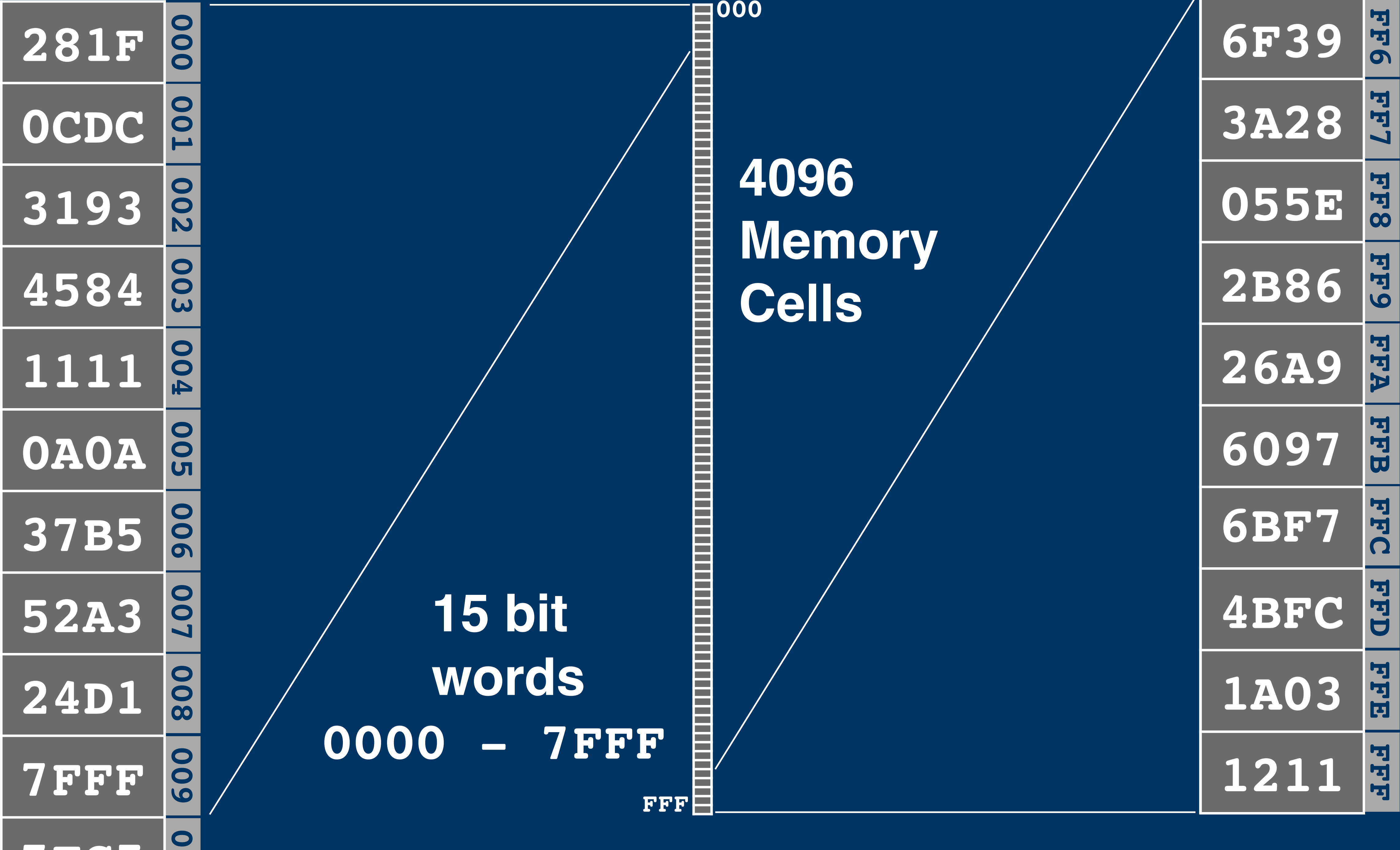


**4096  
Memory  
Cells**

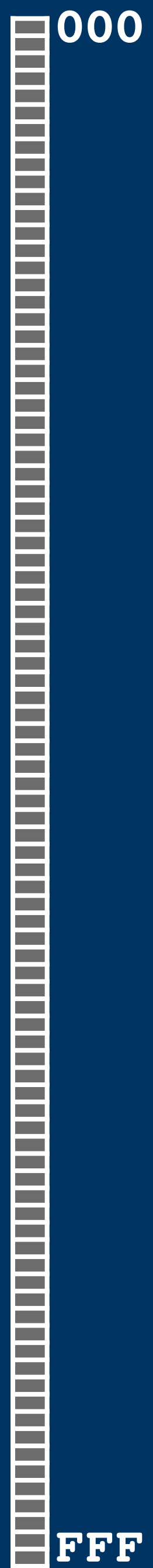














# Accumulator

A

1234

15 bit

000

FFF

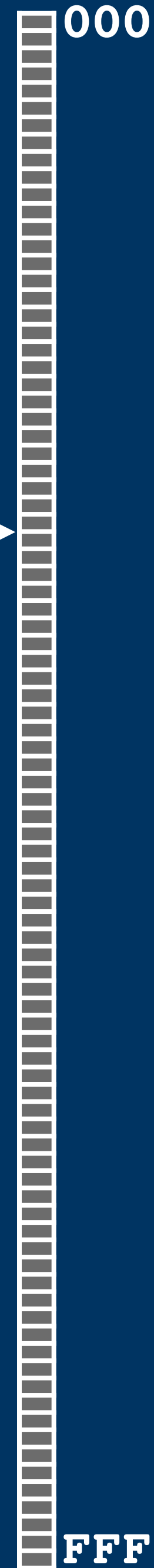


# Accumulator

A

1234

15 bit





# Accumulator

A

1234

15 bit



000

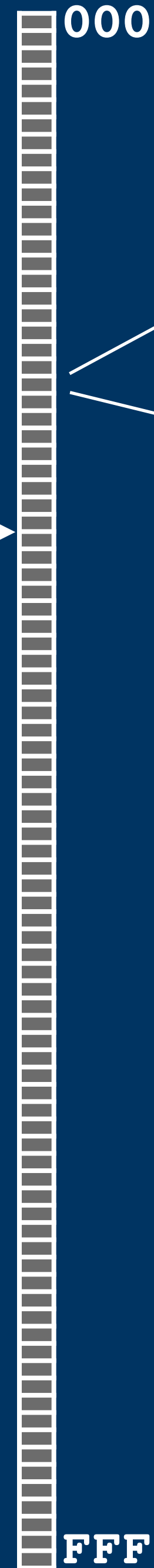
FFF

# Accumulator

A

1234

15 bit



055E

390

0A0A

391

1111

392



# Accumulator

A

1234

15 bit



000

055E

390

0A0A

391

1111

392

```
ld a, [$390]
```

```
add a, [$391]
```

```
ld [$392], a
```

3390

D3A

6391

D3B

5B92

D3E

FFF

# Accumulator

A

1234

15 bit



000

055E

390

0A0A

391

1111

392

```
ld a, [$390]
```

```
add a, [$391]
```

```
ld [$392], a
```

3390

D3A

6391

D3B

5B92

D3E

FFF

von Neumann

single address space  
for code and data



**Accumulator**

**A**

**1234**

**15 bit**



000

**055E**

**390**

**0A0A**

**391**

**1111**

**392**

**ld a, [\$390]**

**3390**

**D3A**

**PC**

**add a, [\$391]**

**6391**

**D3B**

**ld [\$392], a**

**5B92**

**D3E**

FFF

**von Neumann**

**single address space  
for code and data**

**Accumulator**

**A**

**055E**

**15 bit**



000

**055E**

**390**

**0A0A**

**391**

**1111**

**392**

**ld a, [\$390]**

**3390**

**D3A**

**PC**

**add a, [\$391]**

**6391**

**D3B**

**ld [\$392], a**

**5B92**

**D3E**

FFF

**von Neumann**

**single address space  
for code and data**



**Accumulator**

**A**

**0F68**

**15 bit**



000

**055E**

**390**

**0A0A**

**391**

**1111**

**392**

**ld a, [\$390]**

**add a, [\$391]**

**ld [\$392], a**

**3390**

**D3A**

**6391**

**D3B**

**5B92**

**D3E**

**PC**

FFF

**von Neumann**

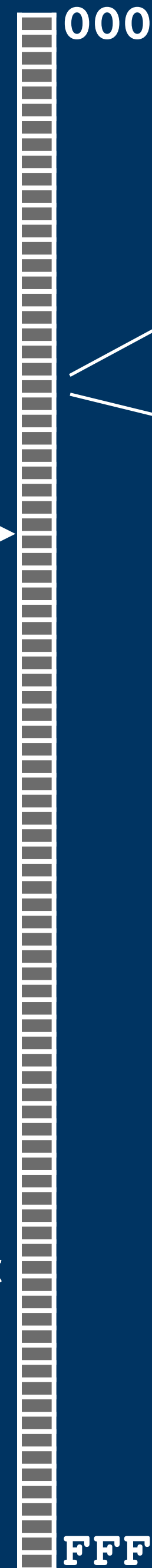
**single address space  
for code and data**

**Accumulator**

**A**

**0F68**

**15 bit**



**055E**

**390**

**0A0A**

**391**

**0F68**

**392**

**ld a, [\$390]**

**add a, [\$391]**

**ld [\$392], a**

**3390**

**D3A**

**6391**

**D3B**

**5B92**

**D3E**

**PC**

**von Neumann**

**single address space  
for code and data**



```
ld a, [$390]  
add a, [$391]  
ld [$392], a
```

```
ld a, [k]
```

```
add a, [k]
```

```
ld [k], a
```



```
ld a, [k]
```

```
add a, [k]
```

```
ld [k], a
```

Load	ld a, [k]	CA k
	3000+k	2
	A ← M[k]	
Add	add a, [k]	AD k
	6000+k	2
	A ← A + M[k]	
Store	ld [k], a	TS k
	5800+k	2
	M[k] ← A	



Load	ld a, [k]	CA k
	3000+k	2
	A ← M[k]	
Add	add a, [k]	AD k
	6000+k	2
	A ← A + M[k]	
Store	ld [k], a	TS k
	5800+k	2
	M[k] ← A	

Load	ld a, [k]	CA k
	3000+k	2
	A ← M[k]	
Add	add a, [k]	AD k
	6000+k	2
	A ← A + M[k]	
Store	ld [k], a	TS k
	5800+k	2
	M[k] ← A	

Syntax





Description

Syntax

Load	ld a, [k]	CA k
	3000+k	2
	A ← M[k]	
Add	add a, [k]	AD k
	6000+k	2
	A ← A + M[k]	
Store	ld [k], a	TS k
	5800+k	2
	M[k] ← A	

Description

Load	ld a, [k]	CA k
	3000+k	2
	A ← M[k]	
Add	add a, [k]	AD k
	6000+k	2
	A ← A + M[k]	
Store	ld [k], a	TS k
	5800+k	2
	M[k] ← A	

Syntax

Operations



Description

Load	ld a, [k]	CA k
	3000+k	2
	A ← M[k]	
Add	add a, [k]	AD k
	6000+k	2
	A ← A + M[k]	
Store	ld [k], a	TS k
	5800+k	2
	M[k] ← A	

Syntax

Encoding

Operations

Description

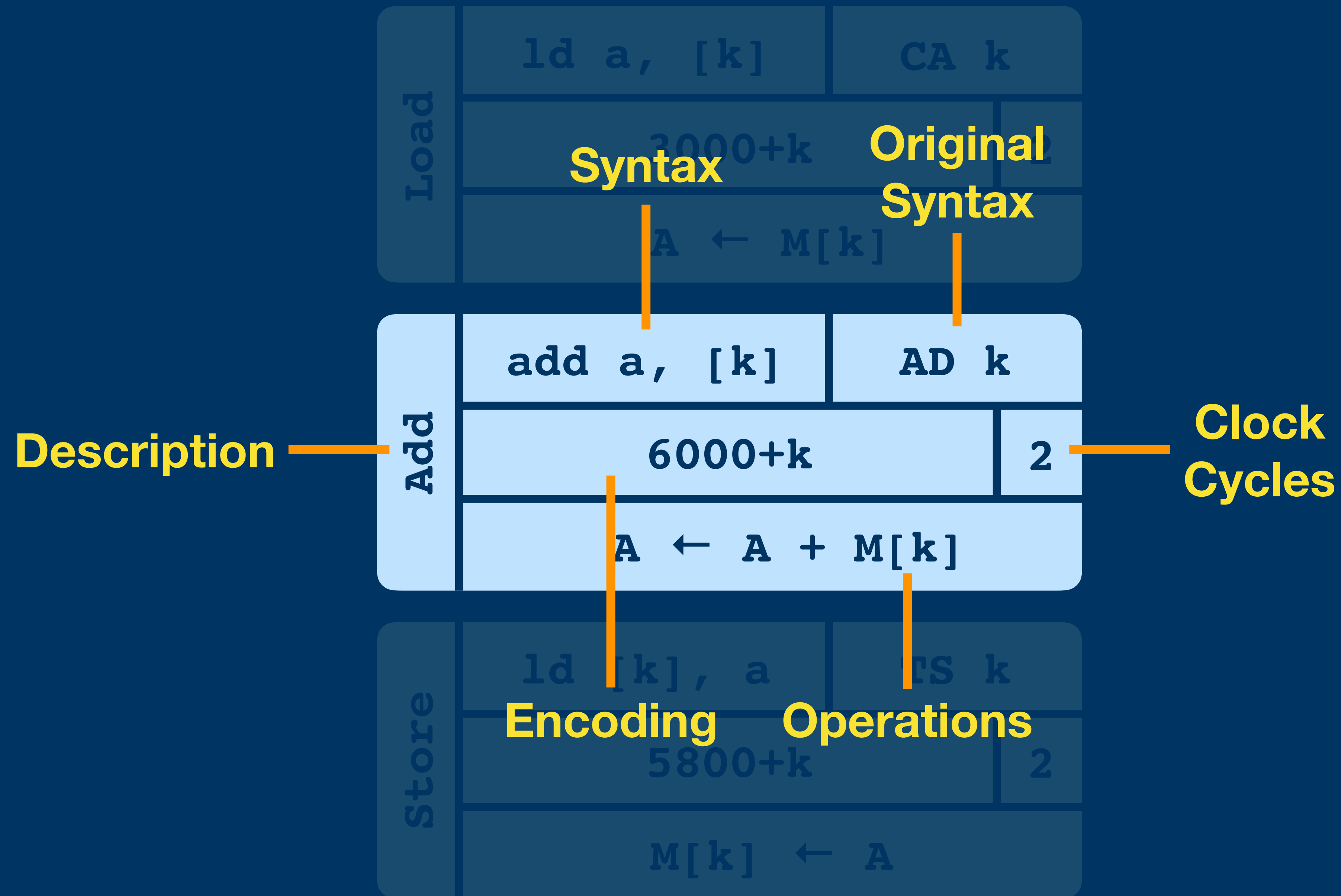
Syntax

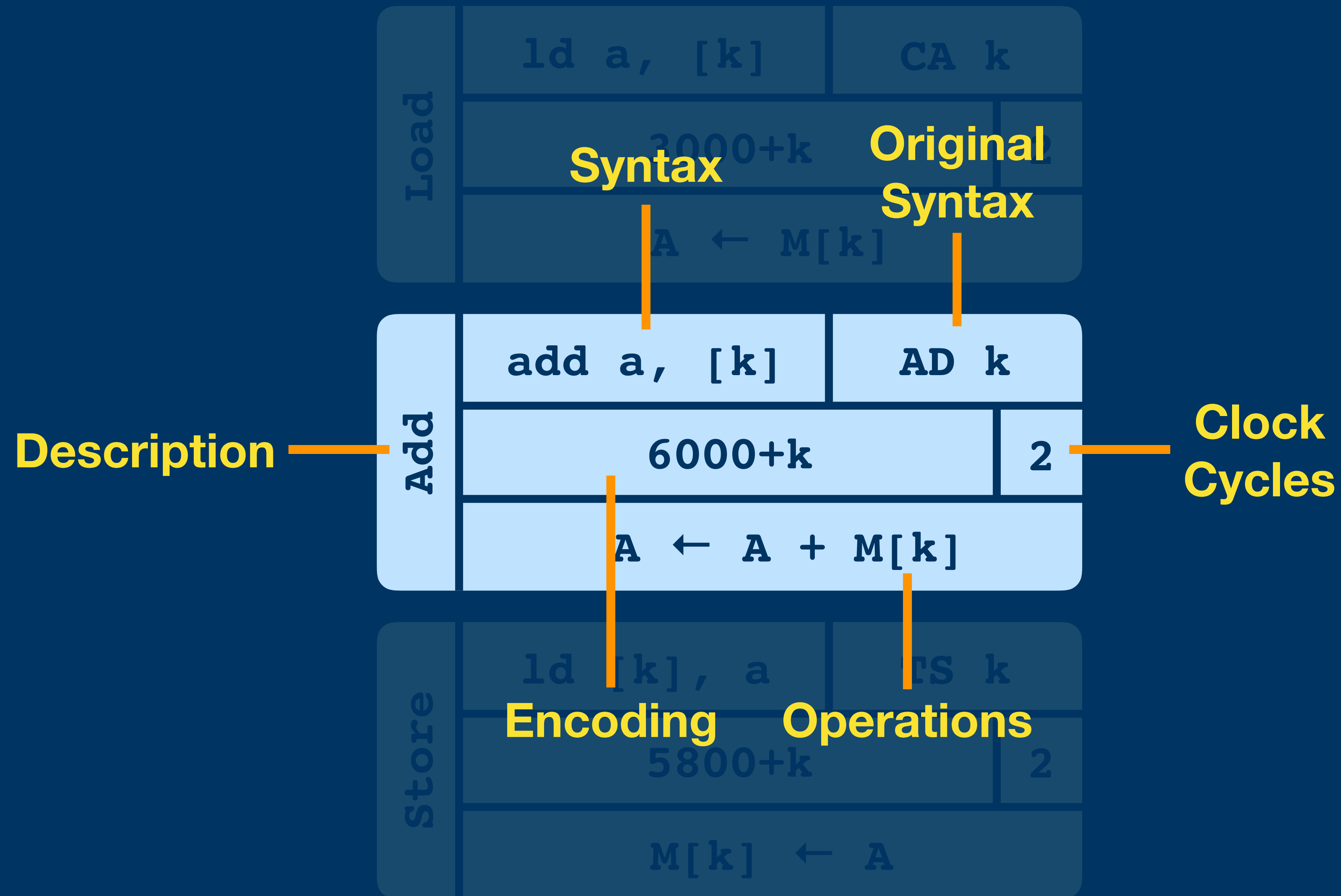
Clock  
Cycles

Encoding Operations

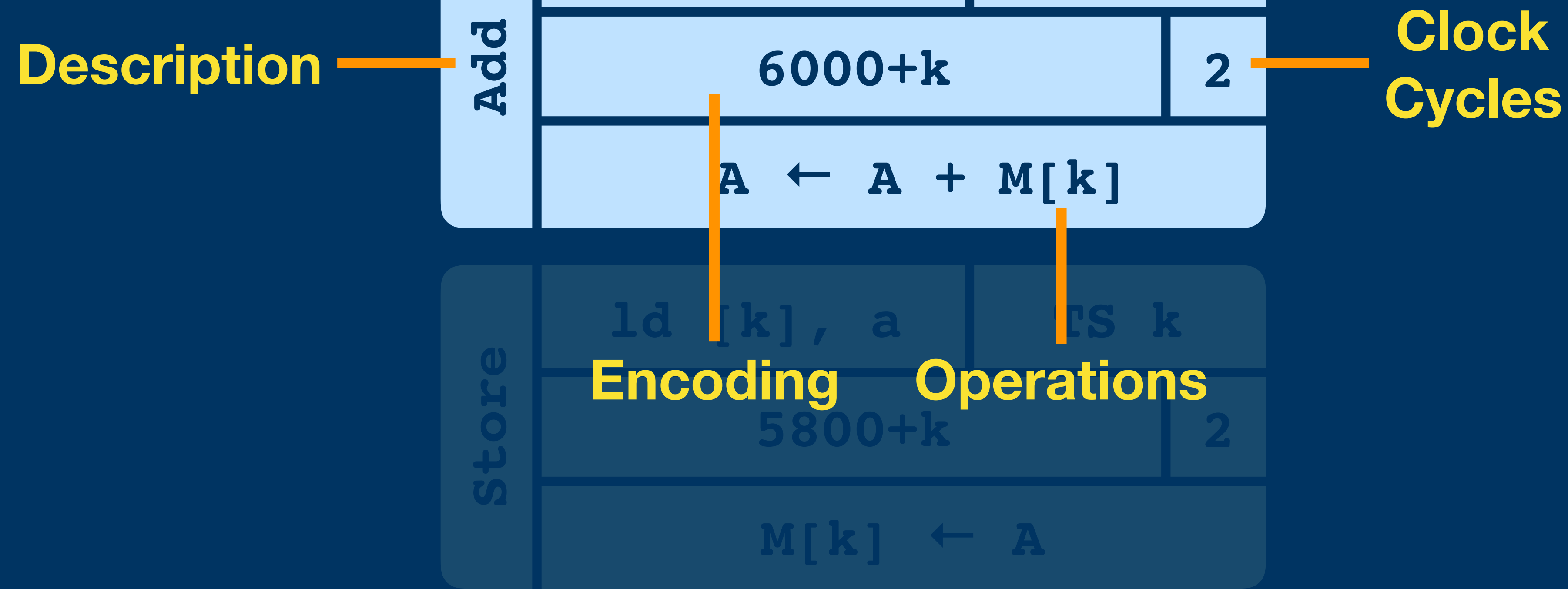
Load	ld a, [k]	CA k
	3000+k	2
	A ← M[k]	
Add	add a, [k]	AD k
	6000+k	2
	A ← A + M[k]	
Store	ld [k], a	TS k
	5800+k	2
	M[k] ← A	











Original	My Syntax
AD	add a, [k]
ADS	add [k], a
CA	ld a, [k]
CCS	ccs
CS	ldc a, [k]
DAS	add [k], ab
DXCH	xchg ab, [k]
INCR	inc k
INHINT	cli
LXCH	xchg b, [k]
MASK	and a, [k]
RELINT	sti
RESUME	iret
TC	call k
TCF	jmp k
TS	ld [k], a
XCH	xchg a, [k]

Load

ld a, [\$200]

Load

ld a, [k]

CA k

3000+k

2

A ← M[k]



Load

ld a, [\$200]

Load	ld a, [k]		CA k	
	3000+k			2
	A ← M[k]			

A	0007
B	0F00
LR	05AA
EB	0000
FB	0000
PC	0421
BB	0000
0	0000

Load

ld a, [\$200]

Load	ld a, [k]		CA k	
	3000+k			2
	A ← M[k]			

A	0007
B	0F00
LR	05AA
EB	0000
FB	0000
PC	0421
BB	0000
0	0000

281F	1FC
0CDC	1FD
3193	1FE
4584	1FF
1111	200
0A0A	201
37B5	202
52A3	203
24D1	204
1111	2



Load

ld a, [\$200]

Load	ld a, [k]	CA k
	3000+k	2
	A ← M[k]	

A	0007
B	0F00
LR	05AA
EB	0000
FB	0000
PC	0421
BB	0000
0	0000

281F	1FC
0CDC	1FD
3193	1FE
4584	1FF
1111	200
0A0A	201
37B5	202
52A3	203
24D1	204
1111	2



Load

ld a, [\$200]

A

1111

B

0F00

LR

05AA

EB

0000

FB

0000

PC

0421

BB

0000

0

0000

281F

1FC

0CDC

1FD

3193

1FE

4584

1FF

1111

200

0A0A

201

37B5

202

52A3

203

24D1

204

2

Load

ld a, [k]

CA k

3000+k

2

A ← M[k]



Store

ld [\$200], a

Store	ld [k], a		TS k	
	5800+k			2
	M[k] ← A			

A	0007
B	0F00
LR	05AA
EB	0000
FB	0000
PC	0421
BB	0000
0	0000

Store

ld [\$200], a

A	0007
B	0F00
LR	05AA
EB	0000
FB	0000
PC	0421
BB	0000
0	0000

281F	1FC
0CDC	1FD
3193	1FE
4584	1FF
0007	200
0A0A	201
37B5	202
52A3	203
24D1	204
1455	205

Store	ld [k], a	TS k
	5800+k	2
	M[k] ← A	



Exchange

xchg a, [\$200]

Exchange	xchg a, [k]		XCH k	
	5C00+k			2
	M[k] ↔ A			

A	0007
B	0F00
LR	05AA
EB	0000
FB	0000
PC	0421
BB	0000
0	0000

Exchange

xchg a, [\$200]

A	1111
B	0F00
LR	05AA
EB	0000
FB	0000
PC	0421
BB	0000
0	0000

281F	1FC
0CDC	1FD
3193	1FE
4584	1FF
0007	200
0A0A	201
37B5	202
52A3	203
24D1	204
1555	205

Exchange	xchg a, [k]	XCH k	
	5C00+k		2
	M[k] ↔ A		



Add

add a, [\$200]

Add	add a, [k]		AD k	
	6000+k			2
	A ← A + M[k]			

A	0007
B	0F00
LR	05AA
EB	0000
FB	0000
PC	0421
BB	0000
0	0000

Add

add a, [\$200]

A

1118

B

0F00

LR

05AA

EB

0000

FB

0000

PC

0421

BB

0000

0

0000



281F

1FC

0CDC

1FD

3193

1FE

4584

1FF

1111

200

0A0A

201

37B5

202

52A3

203

24D1

204

2

Add

add a, [k]

AD k

6000+k

2

A ← A + M[k]



Subtract

sub a, [\$200]

Subtract	sub a, [k]		SU k
	0006, 6000+k		3
	A ← A - M[k]		

A	0007
B	0F00
LR	05AA
EB	0000
FB	0000
PC	0421
BB	0000
0	0000

Subtract

sub a, [\$200]

A	0002
B	0F00
LR	05AA
EB	0000
FB	0000
PC	0421
BB	0000
0	0000



281F	1FC
0CDC	1FD
3193	1FE
4584	1FF
0005	200
0A0A	201
37B5	202
52A3	203
24D1	204
1555	205

Subtract	sub a, [k]	SU k
	0006, 6000+k	3
	A ← A - M[k]	



Subtract

sub a, [\$200]

A	0002
B	0F00
LR	05AA
EB	0000
FB	0000
PC	0421
BB	0000
0	0000



281F	1FC
0CDC	1FD
3193	1FE
4584	1FF
0005	200
0A0A	201
37B5	202
52A3	203
24D1	204
1555	205

Subtract

sub a, [k]

SU k

0006, 6000+k

3

$A \leftarrow A - M[k]$



Negative Numbers

# Negative Numbers



## Negative Numbers

### Unsigned

0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15

# Negative Numbers

## Unsigned

0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15

## Sign-and-value

0000	+0
0001	+1
0010	+2
0011	+3
0100	+4
0101	+5
0110	+6
0111	+7
1000	-0
1001	-1
1010	-2
1011	-3
1100	-4
1101	-5
1110	-6
1111	-7



# Negative Numbers

## Unsigned

0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15

## Sign-and-value

0000	+0
0001	+1
0010	+2
0011	+3
0100	+4
0101	+5
0110	+6
0111	+7
1000	-0
1001	-1
1010	-2
1011	-3
1100	-4
1101	-5
1110	-6
1111	-7

# Negative Numbers

## Unsigned

0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15

## Sign-and-value

0000	+0
0001	+1
0010	+2
0011	+3
0100	+4
0101	+5
0110	+6
0111	+7
1000	-0
1001	-1
1010	-2
1011	-3
1100	-4
1101	-5
1110	-6
1111	-7



# Negative Numbers

## Unsigned

0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15

## Sign-and-value

0000	+0
0001	+1
0010	+2
0011	+3
0100	+4
0101	+5
0110	+6
0111	+7
1000	-0
1001	-1
1010	-2
1011	-3
1100	-4
1101	-5
1110	-6
1111	-7

## One's Complement

0000	+0
0001	+1
0010	+2
0011	+3
0100	+4
0101	+5
0110	+6
0111	+7
1000	-7
1001	-6
1010	-5
1011	-4
1100	-3
1101	-2
1110	-1
1111	-0



# Negative Numbers

## Unsigned

0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15

## Sign-and-value

0000	+0
0001	+1
0010	+2
0011	+3
0100	+4
0101	+5
0110	+6
0111	+7
1000	-0
1001	-1
1010	-2
1011	-3
1100	-4
1101	-5
1110	-6
1111	-7

## One's Complement

0000	+0
0001	+1
0010	+2
0011	+3
0100	+4
0101	+5
0110	+6
0111	+7
1000	-7
1001	-6
1010	-5
1011	-4
1100	-3
1101	-2
1110	-1
1111	-0





# Negative Numbers

## Unsigned

0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15

## Sign-and-value

0000	+0
0001	+1
0010	+2
0011	+3
0100	+4
0101	+5
0110	+6
0111	+7
1000	-0
1001	-1
1010	-2
1011	-3
1100	-4
1101	-5
1110	-6
1111	-7

## One's Complement

0000	+0
0001	+1
0010	+2
0011	+3
0100	+4
0101	+5
0110	+6
0111	+7
1000	-7
1001	-6
1010	-5
1011	-4
1100	-3
1101	-2
1110	-1
1111	-0

## Two's Complement

0000	+0
0001	+1
0010	+2
0011	+3
0100	+4
0101	+5
0110	+6
0111	+7
1000	-8
1001	-7
1010	-6
1011	-5
1100	-4
1101	-3
1110	-2
1111	-1



# Negative Numbers

## Unsigned

0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15

## Sign-and-value

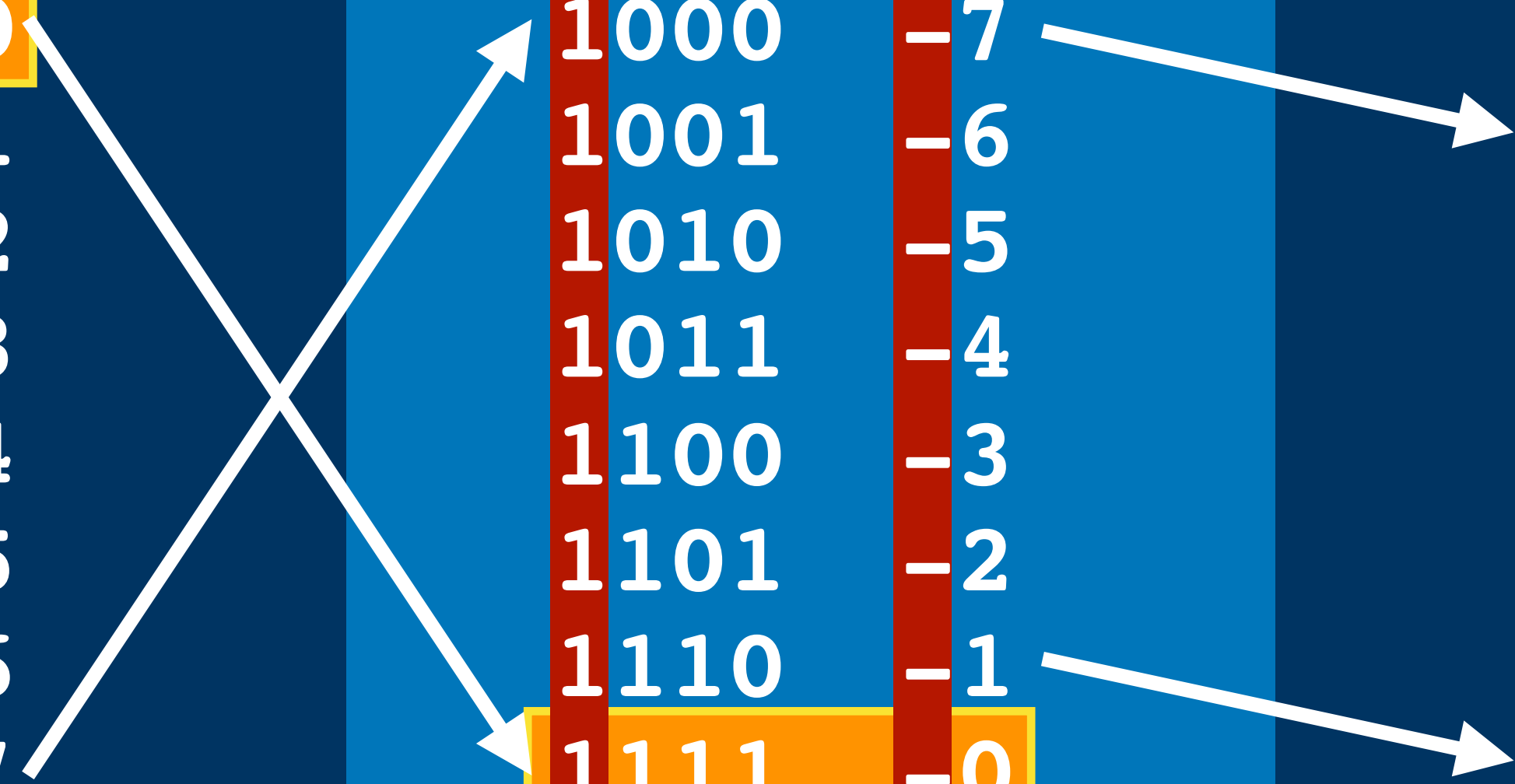
0000	+0
0001	+1
0010	+2
0011	+3
0100	+4
0101	+5
0110	+6
0111	+7
1000	-0
1001	-1
1010	-2
1011	-3
1100	-4
1101	-5
1110	-6
1111	-7

## One's Complement

0000	+0
0001	+1
0010	+2
0011	+3
0100	+4
0101	+5
0110	+6
0111	+7
1000	-7
1001	-6
1010	-5
1011	-4
1100	-3
1101	-2
1110	-1
1111	-0

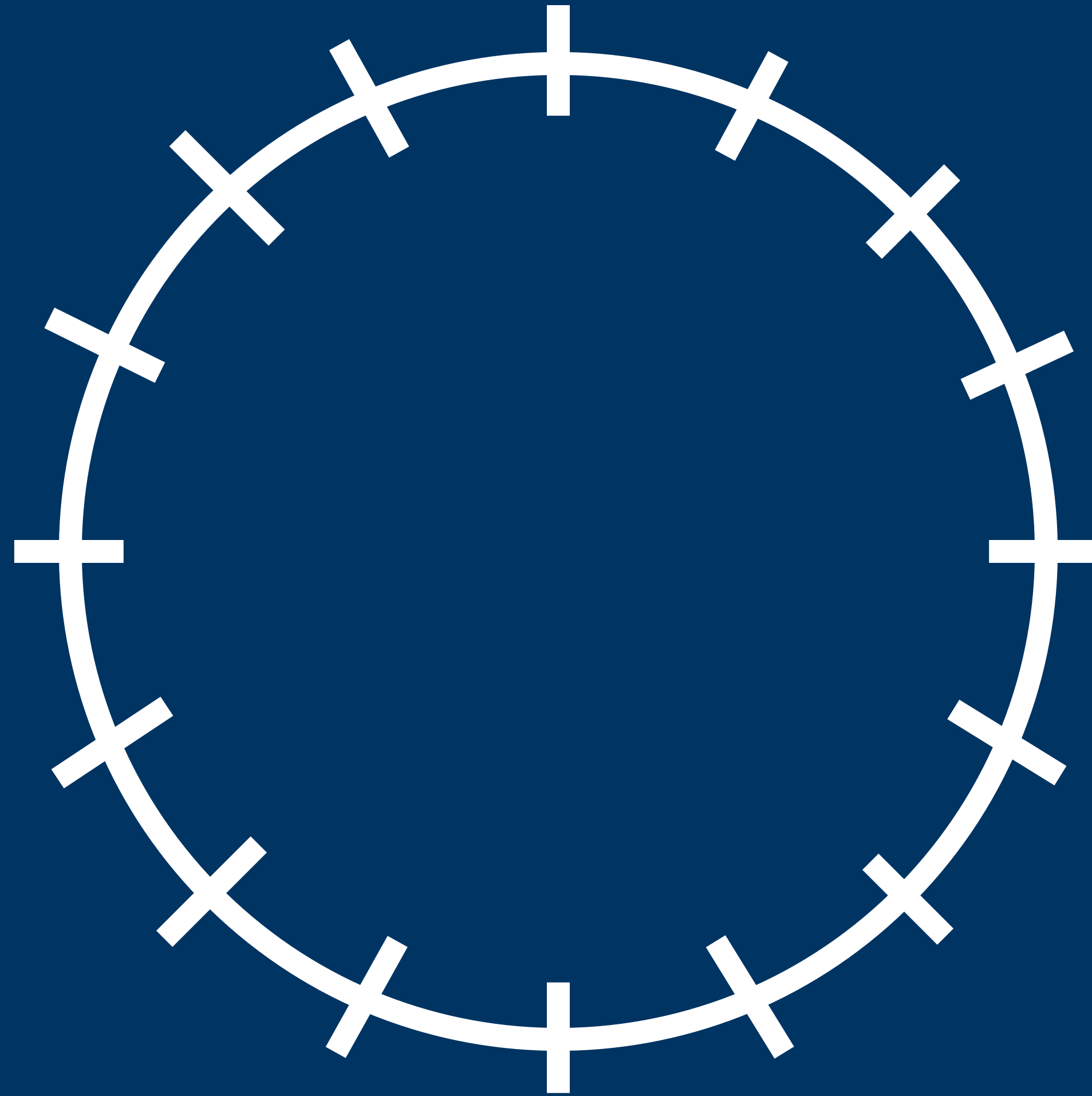
## Two's Complement

0000	+0
0001	+1
0010	+2
0011	+3
0100	+4
0101	+5
0110	+6
0111	+7
1000	-8
1001	-7
1010	-6
1011	-5
1100	-4
1101	-3
1110	-2
1111	-1





# One's Complement



# One's Complement

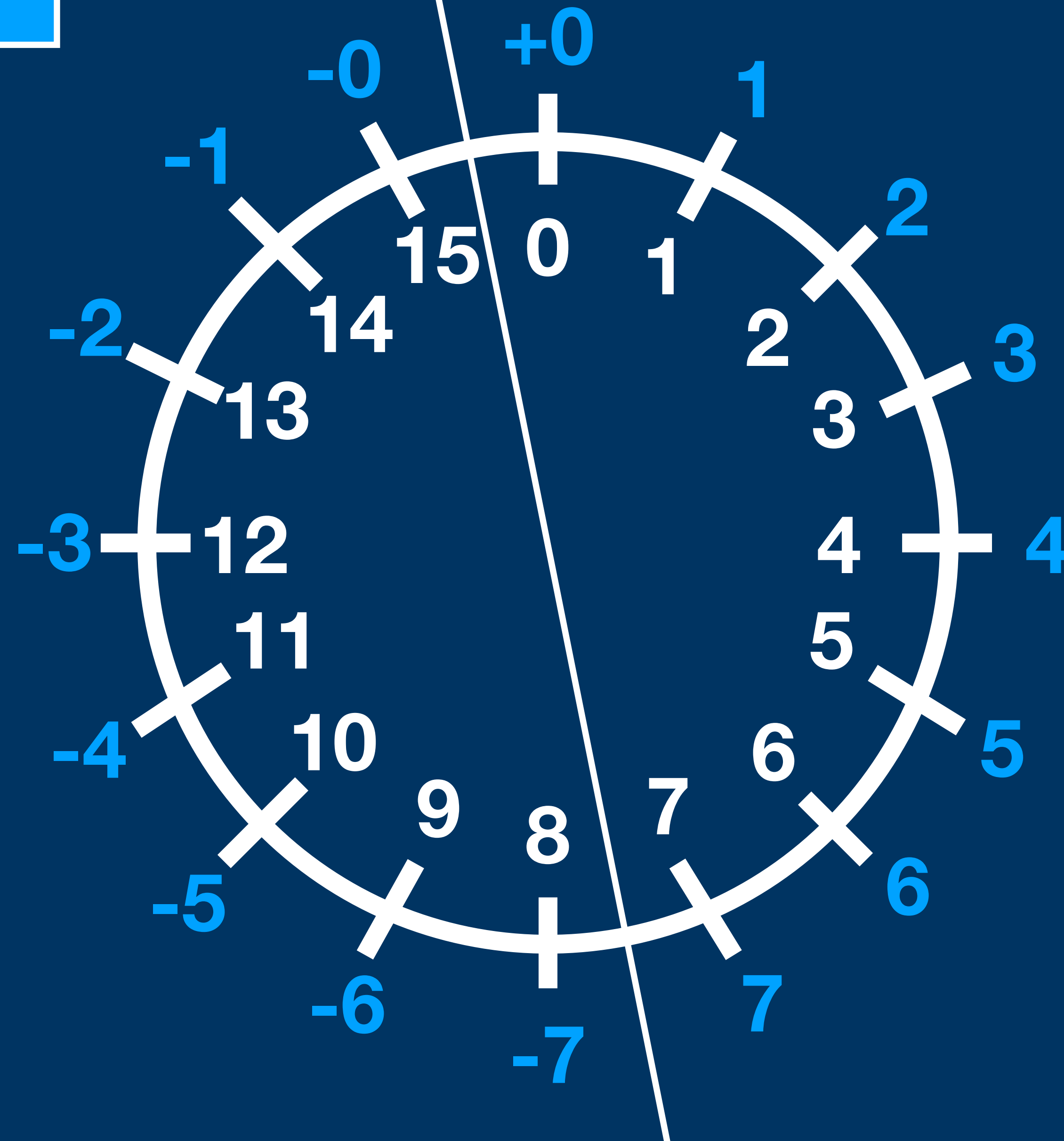




# One's Complement



# One's Complement





# One's Complement



One's Complement

Negate





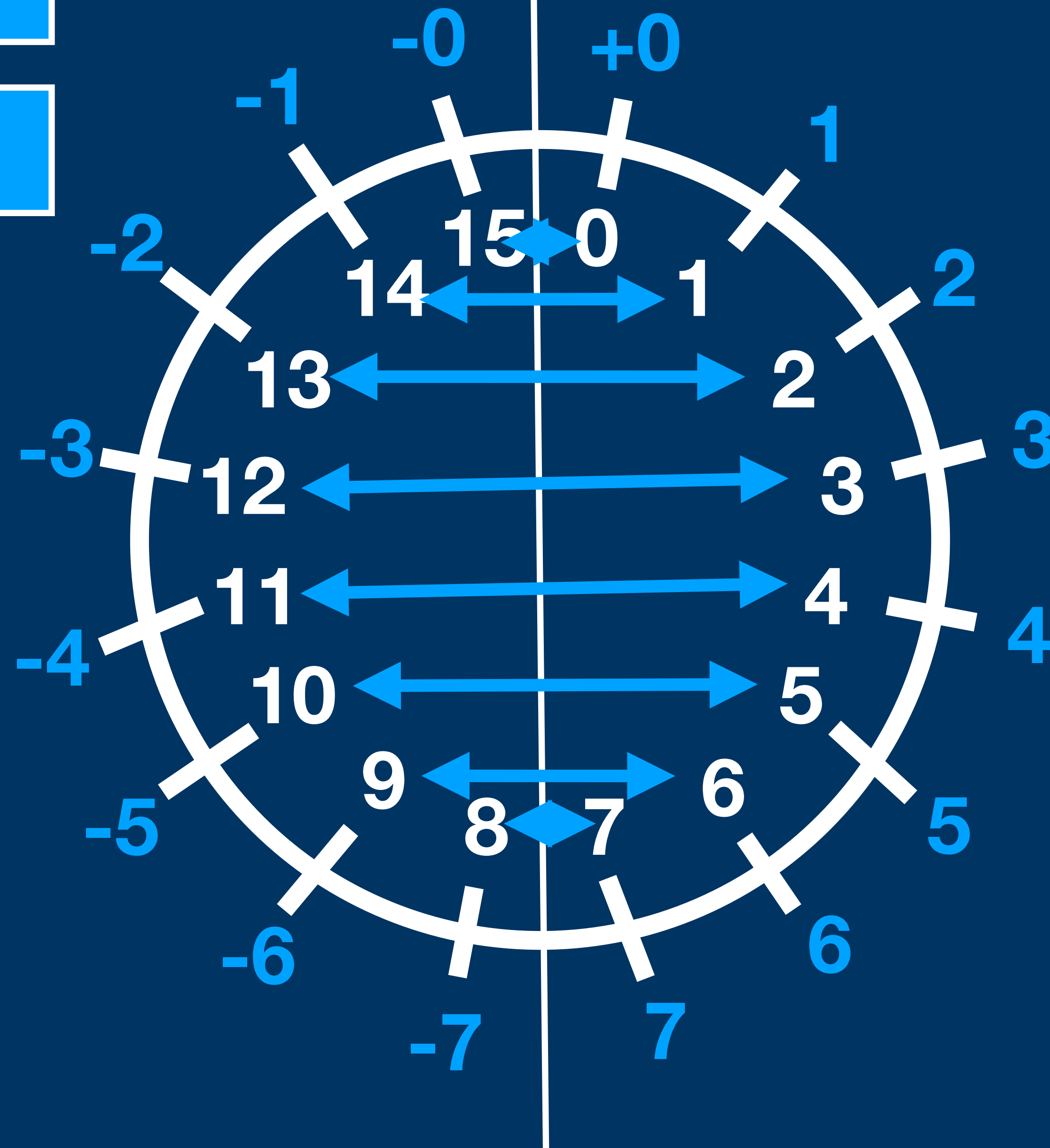
## One's Complement

**Negate**



One's Complement

Negate



+0	0000	↔	1111	-0
+1	0001	↔	1110	-1
+2	0010	↔	1101	-2
+3	0011	↔	1100	-3
+4	0100	↔	1011	-4
+5	0101	↔	1010	-5
+6	0110	↔	1011	-6
+7	0111	↔	1000	-7



One's Complement

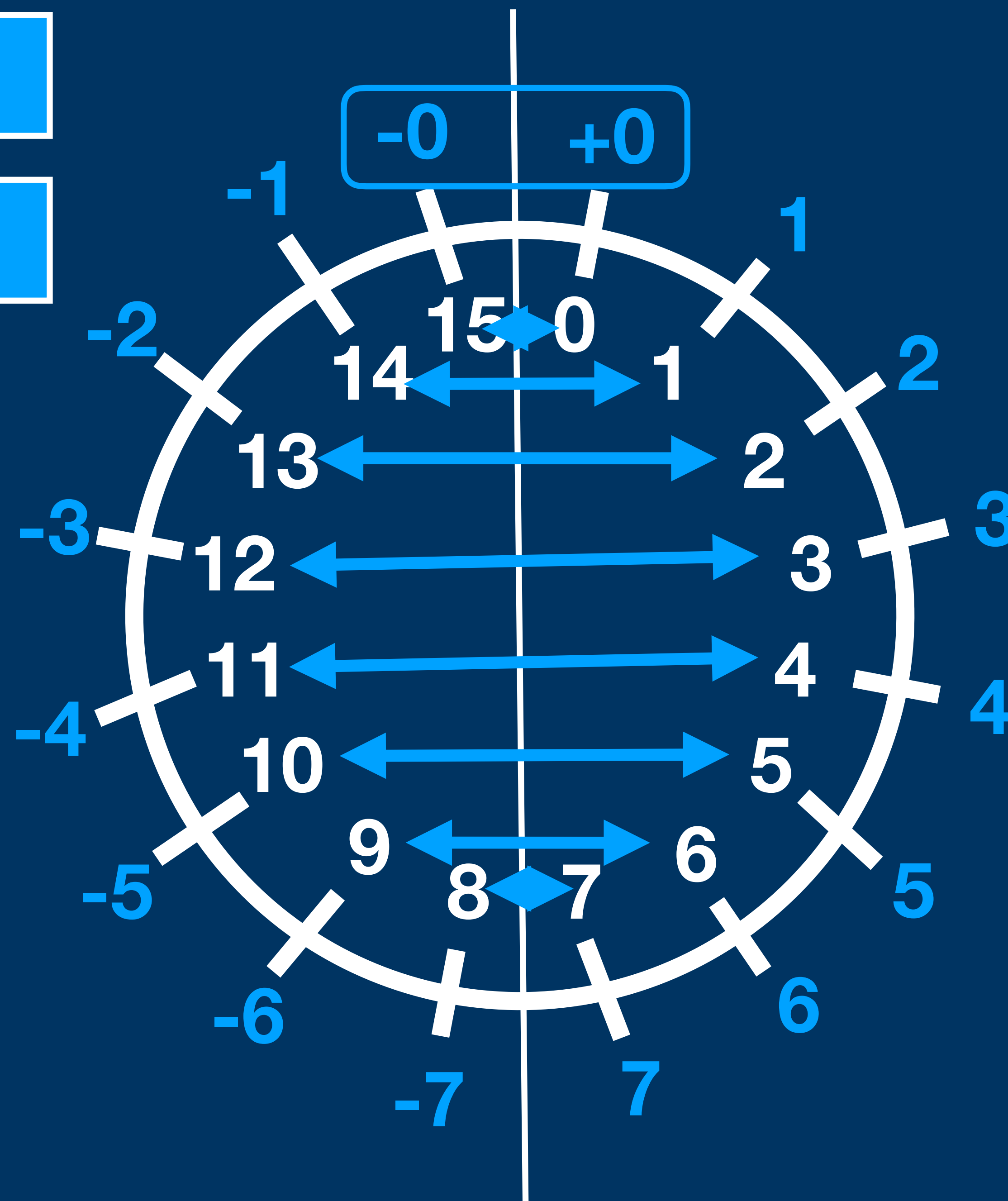
Negate



+0	0000	↔	1111	-0
+1	0001	↔	1110	-1
+2	0010	↔	1101	-2
+3	0011	↔	1100	-3
+4	0100	↔	1011	-4
+5	0101	↔	1010	-5
+6	0110	↔	1011	-6
+7	0111	↔	1000	-7

One's Complement

Negate



+0	0000	↔	1111	-0
+1	0001	↔	1110	-1
+2	0010	↔	1101	-2
+3	0011	↔	1100	-3
+4	0100	↔	1011	-4
+5	0101	↔	1010	-5
+6	0110	↔	1011	-6
+7	0111	↔	1000	-7



One's Complement

Add



One's Complement

Add



$$\begin{aligned} &= 1 + 2 \\ &= 3 \\ &= 3 \end{aligned}$$



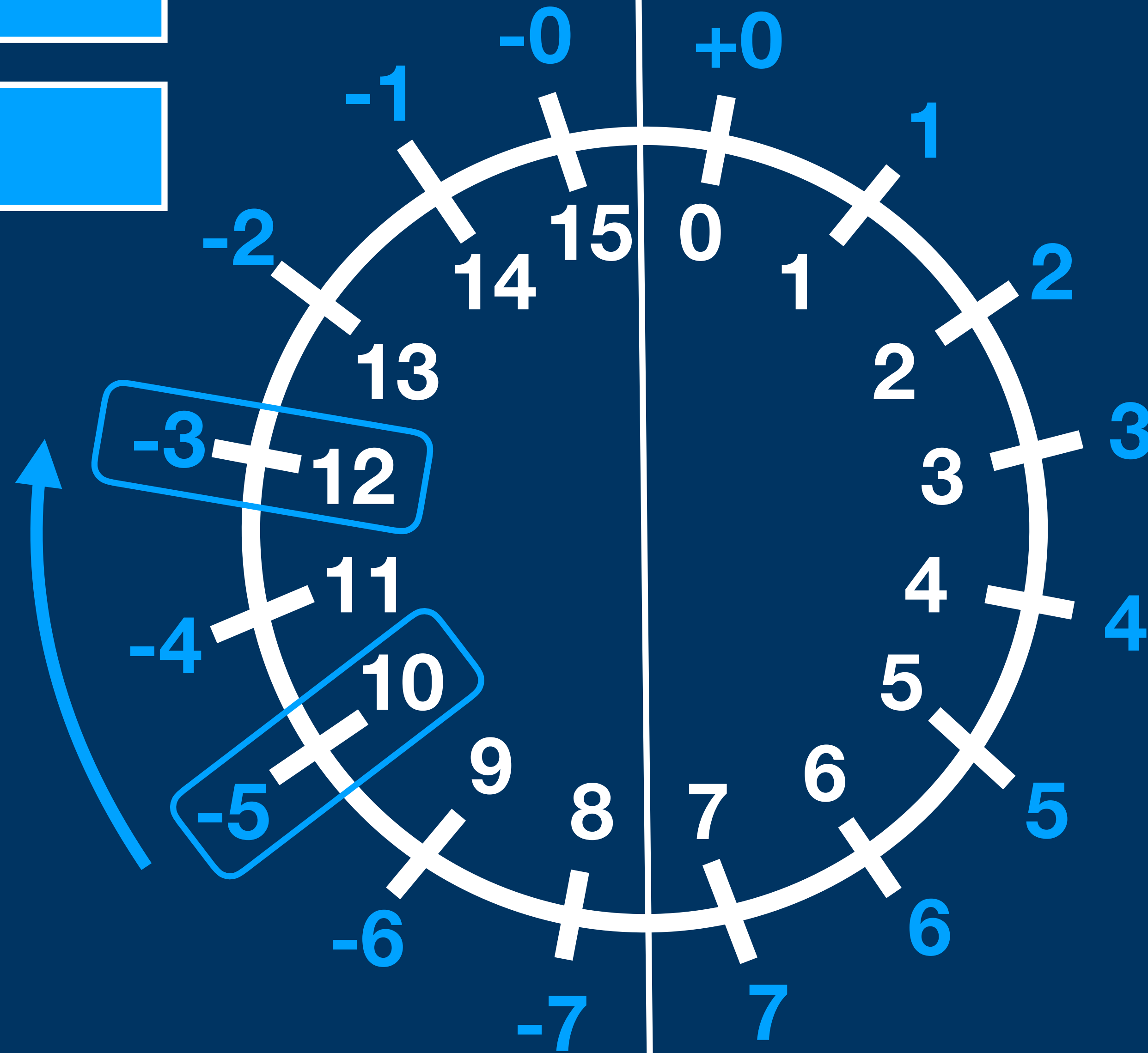
One's Complement

Add



One's Complement

Add



$$\begin{aligned} &= -5 + 2 \\ &= 10 + 2 \\ &= -3 \end{aligned}$$



One's Complement

Subtract



One's Complement

Subtract



$$6 - 4$$



One's Complement

Subtract



$$\begin{array}{r} 6 - 4 \\ = 6 + (-4) \end{array}$$

One's Complement

Subtract



$$\begin{aligned} & 6 - 4 \\ &= 6 + (-4) \\ &= 6 + 11 \end{aligned}$$



One's Complement

Subtract



$$\begin{aligned} & 6 - 4 \\ &= 6 + (-4) \\ &= 6 + 11 \end{aligned}$$

One's Complement

Subtract



$$\begin{aligned} & 6 - 4 \\ &= 6 + (-4) \\ &= 6 + 11 \\ &= 1 \text{Carry} \end{aligned}$$



One's Complement

Subtract



$$\begin{aligned} & 6 - 4 \\ &= 6 + (-4) \\ &= 6 + 11 \\ &= 1 \text{ Carry} \\ &= 2 \end{aligned}$$

One's Complement

Subtract



$$\begin{aligned} & 6 - 4 \\ &= 6 + (-4) \\ &= 6 + 11 \\ &= 1 \text{ Carry} \\ &= 2 \end{aligned}$$

End-Around-Carry



One's Complement

Overflow



One's Complement

Overflow



$$\begin{aligned} &= 7 + 1 \\ &= 8 \\ &= -7 \end{aligned}$$

Positive  
Overflow



# One's Complement



## One's Complement



$$\begin{aligned} &= -7 - 1 \\ &= -7 + (-1) \\ &= 8 + 14 \\ &= 6 \text{ Carry} \\ &= 7 \end{aligned}$$

Negative Overflow



## One's Complement

A 0007

B 0F00

LR 05AA

EB 0000

FB 0000

PC 0421

BB 0000

0 0000



$$\begin{aligned} &= -7 - 1 \\ &= -7 + (-1) \\ &= 8 + 14 \\ &= 6 \text{ Carry} \\ &= 7 \end{aligned}$$

Negative Overflow

# One's Complement

oVerflow

A 0007

B 0F00

LR 05AA

EB 0000

FB 0000

PC 0421

BB 0000

0 0000



$$\begin{aligned} &= -7 - 1 \\ &= -7 + (-1) \\ &= 8 + 14 \\ &= 6 \text{Carry} \\ &= 7 \end{aligned}$$



# One's Complement

A 0007

B 0F00

LR 05AA

EB 0000

FB 0000

PC 0421

BB 0000

0 0000

V



$$\begin{aligned} &= -7 - 1 \\ &= -7 + (-1) \\ &= 8 + 14 \\ &= 6 \text{ Carry} \\ &= 7 \end{aligned}$$

## Overflow Handling

<b>A</b>	<b>0007</b>
<b>B</b>	<b>0F00</b>
<b>LR</b>	<b>05AA</b>
<b>EB</b>	<b>0000</b>
<b>FB</b>	<b>0000</b>
<b>PC</b>	<b>0421</b>
<b>BB</b>	<b>0000</b>
<b>0</b>	<b>0000</b>

```
ld a, [$200]
```

```
add a, [$201]
```

```
ld [$203], a
```

```
jmp cont
```

```
ld [$202], a
```

# Overflow Handling

A	7FFF
B	0F00
LR	05AA
EB	0000
FB	0000
PC	0421
BB	0000
0	0000

PC →

```
ld a, [$200]
```

```
add a, [$201]
```

```
ld [$203], a
```

```
jmp cont
```

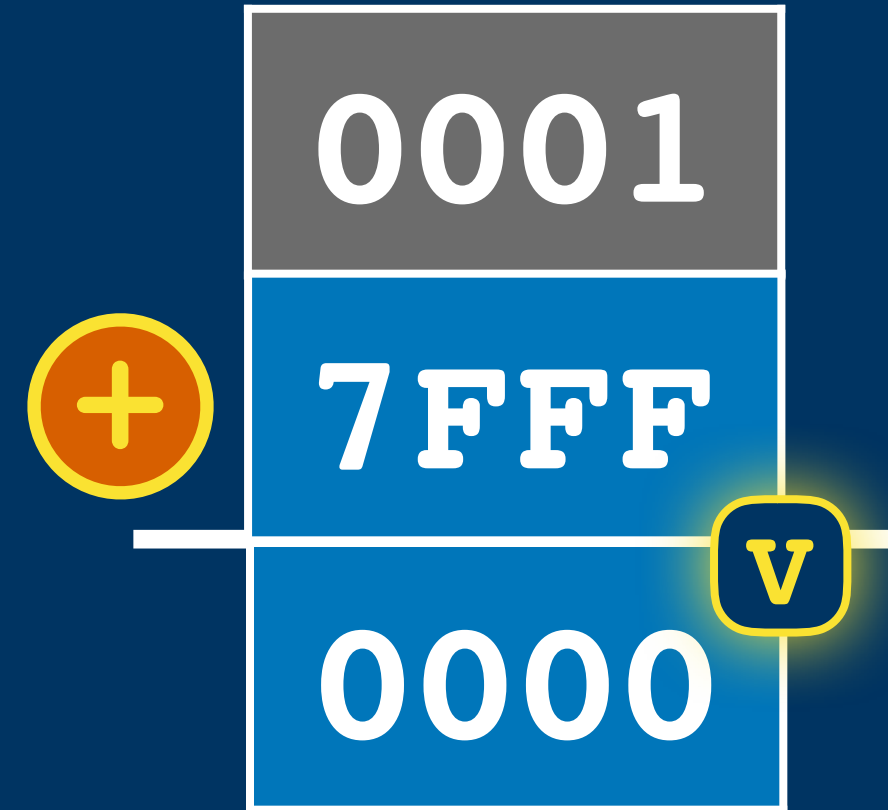
```
ld [$202], a
```

4584	FF
7FFF	200
0001	201
37B5	202
74FE	203
24D1	204



# Overflow Handling

A	7FFF
B	0F00
LR	05AA
EB	0000
FB	0000
PC	0421
BB	0000
0	0000



PC →

```
ld a, [$200]
add a, [$201]
ld [$203], a
jmp cont
ld [$202], a
```

4584	FF
7FFF	200
0001	201
37B5	202
74FE	203
24D1	204

# Overflow Handling

A	0000
B	0F00
LR	05AA
EB	0000
FB	0000
PC	0421
BB	0000
0	0000

V



PC →

```
ld a, [$200]
add a, [$201]
ld [$203], a
jmp cont
ld [$202], a
```

4584	FF
7FFF	200
0001	201
37B5	202
74FE	203
24D1	204

# Overflow Handling

A	0000
B	0F00
LR	05AA
EB	0000
FB	0000
PC	0421
BB	0000
0	0000

V



ld a, [\$200]  
add a, [\$201]  
PC → ld [\$203], a  
jmp cont  
ld [\$202], a

4584	FF
7FFF	200
0001	201
37B5	202
74FE	203
24D1	204



# Overflow Handling

A	0000
B	0F00
LR	05AA
EB	0000
FB	0000
PC	0421
BB	0000
0	0000

V



PC →

ld a, [\$200]

add a, [\$201]

ld [\$203], a

jmp cont

ld [\$202], a

7FFF	200
0001	201
37B5	202
0000	203
24D1	204

# Overflow Handling

A	0000
B	0F00
LR	05AA
EB	0000
FB	0000
PC	0421
BB	0000
0	0000



ld a, [\$200]  
add a, [\$201]  
PC → ld [\$203], a  
jmp cont  
ld [\$202], a

FF	4584
200	7FFF
201	0001
202	37B5
203	0000
204	24D1

# Overflow Handling

A	0001
B	0F00
LR	05AA
EB	0000
FB	0000
PC	0421
BB	0000
0	0000



ld a, [\$200]  
add a, [\$201]  
PC → ld [\$203], a  
jmp cont  
ld [\$202], a

FF	4584
200	7FFF
201	0001
202	37B5
203	0000
204	24D1



# Overflow Handling

A	0001
B	0F00
LR	05AA
EB	0000
FB	0000
PC	0421
BB	0000
0	0000



```
ld a, [$200]
add a, [$201]
ld [$203], a
jmp cont
```

PC → ld [\$202], a

FF	4584
200	7FFF
201	0001
202	37B5
203	0000
20	24D1

# Overflow Handling

A	0001
B	0F00
LR	05AA
EB	0000
FB	0000
PC	0421
BB	0000
0	0000



```
ld a, [$200]
add a, [$201]
ld [$203], a
jmp cont
ld [$202], a
```

4584	FF	
7FFF	200	
0001	201	
0001	202	Double Word Result
0000	203	
24D1	204	

Load Complement

ldc a, [\$200]

Load Cpl	ldc a, [k]		CS k	
	4000+k			2
	A ← -M[k]			

A	0007
B	0F00
LR	05AA
EB	0000
FB	0000
Z	0421
BB	0000
0	0000



Load Complement

ldc a, [\$200]

Load Cpl	ldc a, [k]	CS k
	4000+k	2
	A ← -M[k]	

A	0007
B	0F00
LR	05AA
EB	0000
FB	0000
Z	0421
BB	0000
0	0000

1111

281F	1FC
0CDC	1FD
3193	1FE
4584	1FF
1111	200
0A0A	201
37B5	202
52A3	203
24D1	204
1111	2

Load Complement

ldc a, [\$200]

Load Cpl	ldc a, [k]	CS k
	4000+k	2
	A ← -M[k]	

A	0007
B	0F00
LR	05AA
EB	0000
FB	0000
Z	0421
BB	0000
0	0000

6EEE

281F	1FC
0CDC	1FD
3193	1FE
4584	1FF
1111	200
0A0A	201
37B5	202
52A3	203
24D1	204
1555	2

Load Complement

A	6EEE
B	0F00
LR	05AA
EB	0000
FB	0000
Z	0421
BB	0000
0	0000

ldc a, [\$200]

281F	1FC
0CDC	1FD
3193	1FE
4584	1FF
1111	200
0A0A	201
37B5	202
52A3	203
24D1	204
1111	2



Load Cpl	ldc a, [k]	CS k
	4000+k	2
	A ← -M[k]	



Increment

`inc [$200]`

Increment	inc [k]		INCR k	
	2800+k			2
	M[k] ← M[k]+1			

A	0007
B	0F00
LR	05AA
EB	0000
FB	0000
PC	0421
BB	0000
0	0000

Increment

A 0007

B 0F00

LR 05AA

EB 0000

FB 0000

PC 0421

BB 0000

0 0000

inc [\$200]

281F

0CDC

3193

4584

1112

0A0A

37B5

52A3

24D1

1111

1FC

1FD

1FE

1FF

200

201

202

203

204

2

Increment

inc [k]

INCR k

2800+k

2

M[k] ← M[k]+1

Augment/Diminish

Increment	inc [k]	INCR k
	2800+k	2
	M[k] ← M[k]+1	



Augment/Diminish

Increment	inc [k]	INCR k
	2800+k	2
	M[k] ← M[k]+1	

Augment	aug [k]	AUG k
	0006, 2800+k	3
	if M[k] ≥ +0: M[k] ← M[k]+1 if M[j] ≤ -0: M[k] ← M[k]-1	

Diminish	dim [k]	DIM k
	0006, 2C00+k	3
	if M[k] > +0: M[k] ← M[k]-1 if M[j] < -0: M[k] ← M[k]+1	

Inc/Augment/Diminish

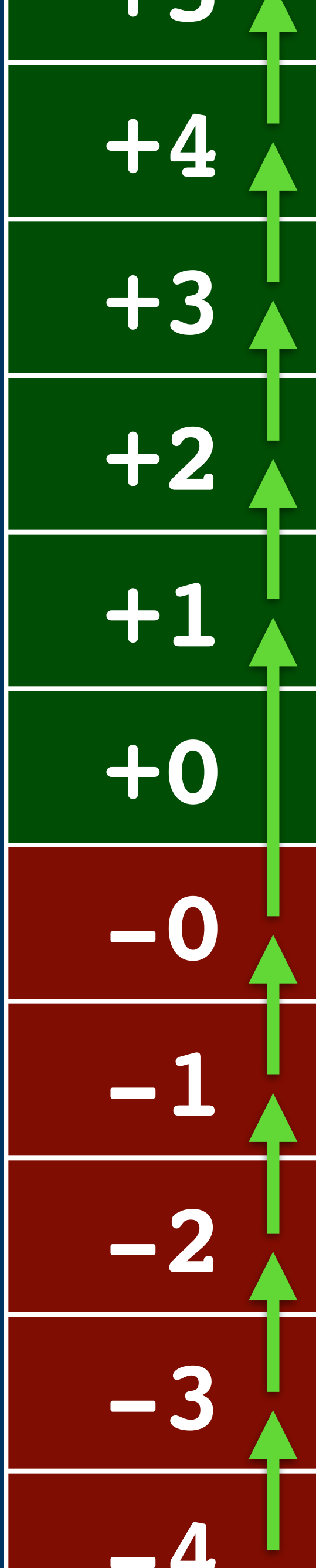
Increment

Augment

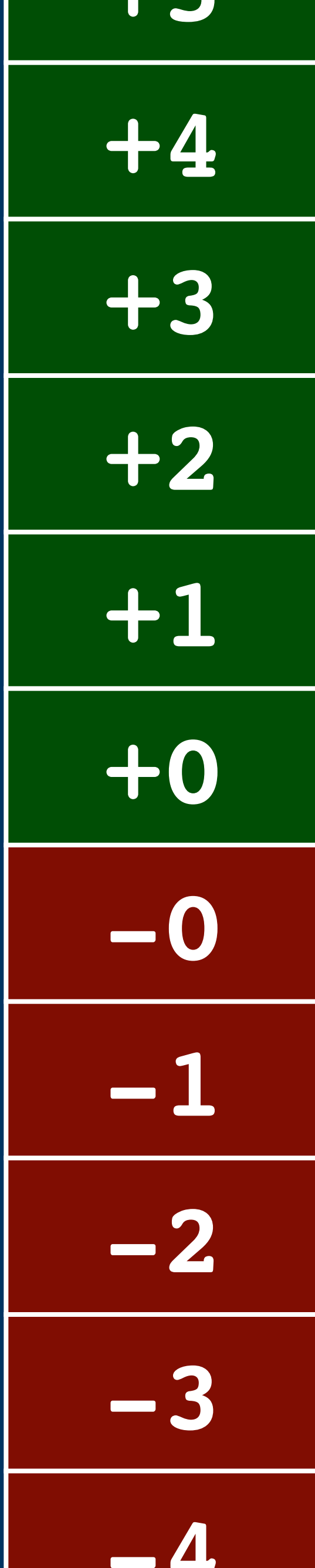
Diminish

Inc/Augment/Diminish

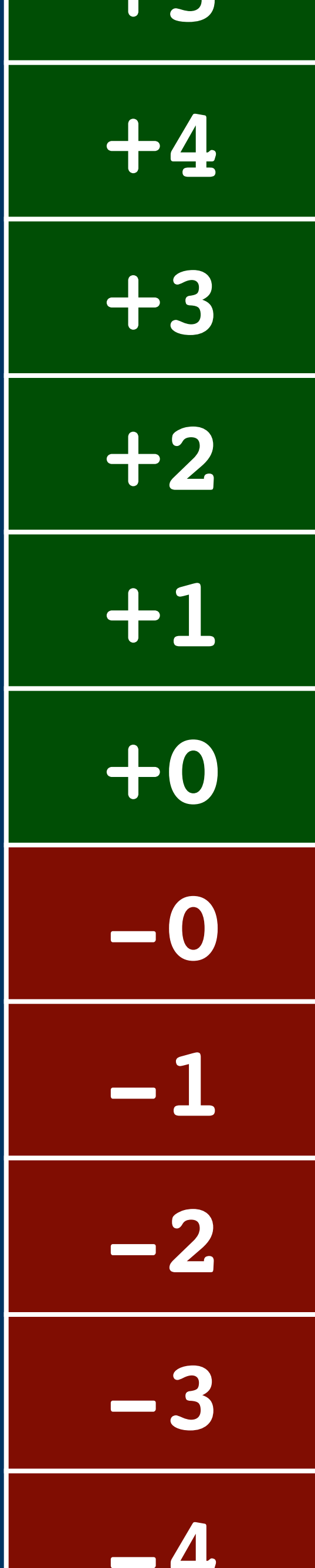
Increment



Augment

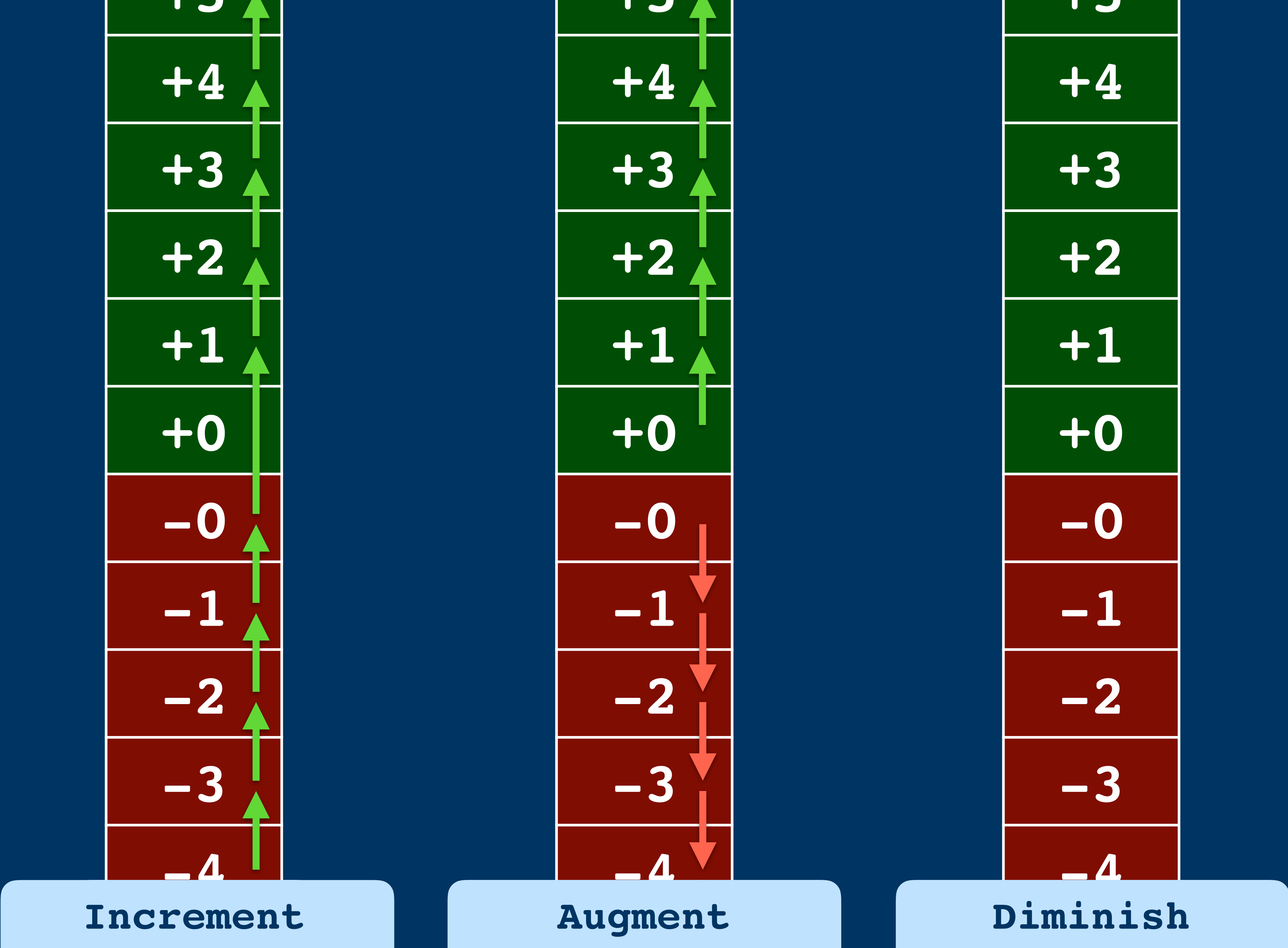


Diminish





Inc/Augment/Diminish



Inc/Augment/Diminish



Multiply

mul [\$200]

A0007

B0F00

LR05AA

EB0000

FB0000

PC0421

BB0000

00000

Multiply

mul [k]

MP k

0006, 7000+k

4

A,B ← A × M[k]



Multiply

mul [\$200]

A	0007
B	0F00
LR	05AA
EB	0000
FB	0000
PC	0421
BB	0000
0	0000



0007
3333
2665

281F	1FC
0CDC	1FD
3193	1FE
4584	1FF
3333	200
0A0A	201
37B5	202
52A3	203
24D1	204
1555	205



Multiply

mul [k]	MP k
0006, 7000+k	4
A, B ← A × M[k]	

Multiply

mul [\$200]

A	0007	0007
B	0F00	3333
LR	05A	0005 2665
EB	0000	
FB	0000	
PC	0421	
BB	0000	
0	0000	



281F	1FC
0CDC	1FD
3193	1FE
4584	1FF
3333	200
0A0A	201
37B5	202
52A3	203
24D1	204
1555	205



Multiply

mul [k]	MP k
0006, 7000+k	4
A, B ← A × M[k]	

Multiply

mul [\$200]

A	0005
B	2665
LR	05AA
EB	0000
FB	0000
PC	0421
BB	0000
0	0000



281F	1FC
0CDC	1FD
3193	1FE
4584	1FF
3333	200
0A0A	201
37B5	202
52A3	203
24D1	204
1555	205

Multiply	mul [k]	MP k
	0006, 7000+k	4
	A, B ← A × M[k]	



## Double Word Values

**A**

**0005**

**B**

**2665**

## Double Word Values

0005

A

2665

B

## Double Word Values

0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	0	1	1	0	0	1	1	0	0	1	0	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

A

B



## Double Word Values

$(\$0005 \ll 14) \mid \$2665$

0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	1	1	0	0	1	1	0	0	1	0	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

A

B

$= \$16665$

Divide

div [\$200]

A	0000
B	0007
LR	05AA
EB	0000
FB	0000
PC	0421
BB	0000
0	0000

Divide	div [k]		DV k
	0006, 1000+k		7
	<div>A ← A, B ÷ M[k]</div> <div>B ← A, B % M[k]</div>		

Divide

div [\$200]

A	0000	0000	0007
B	0007	<div>÷</div>	0003
LR	05AA		0002
EB	0000		
FB	0000		
PC	0421		
BB	0000		
0	0000		

281F	1FC
0CDC	1FD
3193	1FE
4584	1FF
0003	200
0A0A	201
37B5	202
52A3	203
24D1	204
1555	2



Divide	div [k]	DV k
	0006, 1000+k	7
	A ← A, B ÷ M[k] B ← A, B % M[k]	



Divide

div [\$200]

A

0000

0000

0007

B

0007



0003

LR

05AA

0002

0001

584

EB

0000

0003

FB

0000

0A0A

PC

0421

37B5

BB

0000

52A3

0

0000

24D1

281F

0CDC

3193

Remainder

1FC

1FD

1FE

1FF

200

201

202

203

204

2

Divide

div [k]

DV k

0006, 1000+k

7

A ← A, B ÷ M[k]

B ← A, B % M[k]

Divide

div [\$200]

A	0002
B	0001
LR	05AA
EB	0000
FB	0000
PC	0421
BB	0000
0	0000



281F	1FC
0CDC	1FD
3193	1FE
4584	1FF
0003	200
0A0A	201
37B5	202
52A3	203
24D1	204
1555	2

Divide	div [k]	DV k
	0006, 1000+k	7
	A ← A, B ÷ M[k] B ← A, B % M[k]	

Load Double

ld ab, [\$200]

A	0007
B	0F00
LR	05AA
EB	0000
FB	0000
PC	0421
BB	0000
0	0000

Load Double	ld ab, [k]	DCA k
	0006, 3001+k	4
	A ← M[k] B ← M[k+1]	



Load Double

A	0007
B	0F00
LR	05AA
EB	0000
FB	0000
PC	0421
BB	0000
0	0000

ld ab, [\$200]

281F	1FC
0CDC	1FD
3193	1FE
4584	1FF
1111	200
0A0A	201
37B5	202
52A3	203
24D1	204
1111	205



Load Double	ld ab, [k]	DCA k
	0006, 3001+k	4
	A ← M[k] B ← M[k+1]	

Load Double

ld ab, [\$200]

A	1111
B	0A0A
LR	05AA
EB	0000
FB	0000
PC	0421
BB	0000
0	0000

281F	1FC
0CDC	1FD
3193	1FE
4584	1FF
1111	200
0A0A	201
37B5	202
52A3	203
24D1	204
1111	205



Load Double	ld ab, [k]	DCA k
	0006, 3001+k	4
	A ← M[k] B ← M[k+1]	

Load Double

A	1111
B	0A0A
LR	05AA
EB	0000
FB	0000
PC	0421
BB	0000
0	0000

ld ab, [\$200]

281F	1FC
0CDC	1FD
3193	1FE
4584	1FF
1111	200
0A0A	201
37B5	202
52A3	203
24D1	204
1111	205



Load Double	ld ab, [k]	DCA k
	0006, 3001+k	4
	A ← M[k] B ← M[k+1]	
Ld Dbl Cpl	ldc ab, [k]	DCS k
	0006, 4001+k	4
	A ← -M[k] B ← -M[k+1]	



Exchange Double

xchg ab, [\$200]

Xchg Double	xchg ab, [k]		DXCH k	
	5201+k			3
	A ↔ M[k] B ↔ M[k+1]			

A	0007
B	0F00
LR	05AA
EB	0000
FB	0000
PC	0421
BB	0000
0	0000

Exchange Double

xchg ab, [\$200]

A	1111
B	0A0A
LR	05AA
EB	0000
FB	0000
PC	0421
BB	0000
0	0000

281F	1FC
0CDC	1FD
3193	1FE
4584	1FF
0007	200
0F00	201
37B5	202
52A3	203
24D1	204
1555	205



Xchg Double	xchg ab, [k]	DXCH k
	5201+k	
	A ↔ M[k] B ↔ M[k+1]	

Exchange Double

xchg ab, [\$200]

A	1111
B	0A0A
LR	05AA
EB	0000
FB	0000
PC	0421
BB	0000
0	0000

281F	1FC
0CDC	1FD
3193	1FE
4584	1FF
0007	200
0F00	201
37B5	202
52A3	203
24D1	204
1555	205



Xchg Double	xchg ab, [k]	DXCH k	
	5201+k		3
	A ↔ M[k] B ↔ M[k+1]		

Add Double	add [k], ab	DAS k	
	2001+k		3
	M[k,k+1] ← M[k,k+1] + A,B		



Exchange B

xchg b, [\$200]

Exchg B	xchg b, [k]		LXCH k	
	2400+k			2
	M[k] ↔ B			

A	0007
B	0F00
LR	05AA
EB	0000
FB	0000
PC	0421
BB	0000
0	0000

Exchange B

xchg b, [\$200]

A	0007
B	1111
LR	05AA
EB	0000
FB	0000
PC	0421
BB	0000
0	0000

281F	1FC
0CDC	1FD
3193	1FE
4584	1FF
0F00	200
0A0A	201
37B5	202
52A3	203
24D1	204
1111	205

Exchg B	xchg b, [k]	LXCH k	
	2400+k		2
	M[k] ↔ B		

## Indexing

`ld a, [$700+[$80]]`

<b>A</b>	<b>0000</b>
<b>B</b>	<b>0F00</b>
<b>LR</b>	<b>05AA</b>
<b>EB</b>	<b>0000</b>
<b>FB</b>	<b>0000</b>
<b>PC</b>	<b>0421</b>
<b>BB</b>	<b>0000</b>
<b>0</b>	<b>0000</b>



Indexing

ld a, [\$700+[\$80]]

A	0000
B	0F00
LR	05AA
EB	0000
FB	0000
PC	0421
BB	0000
0	0000

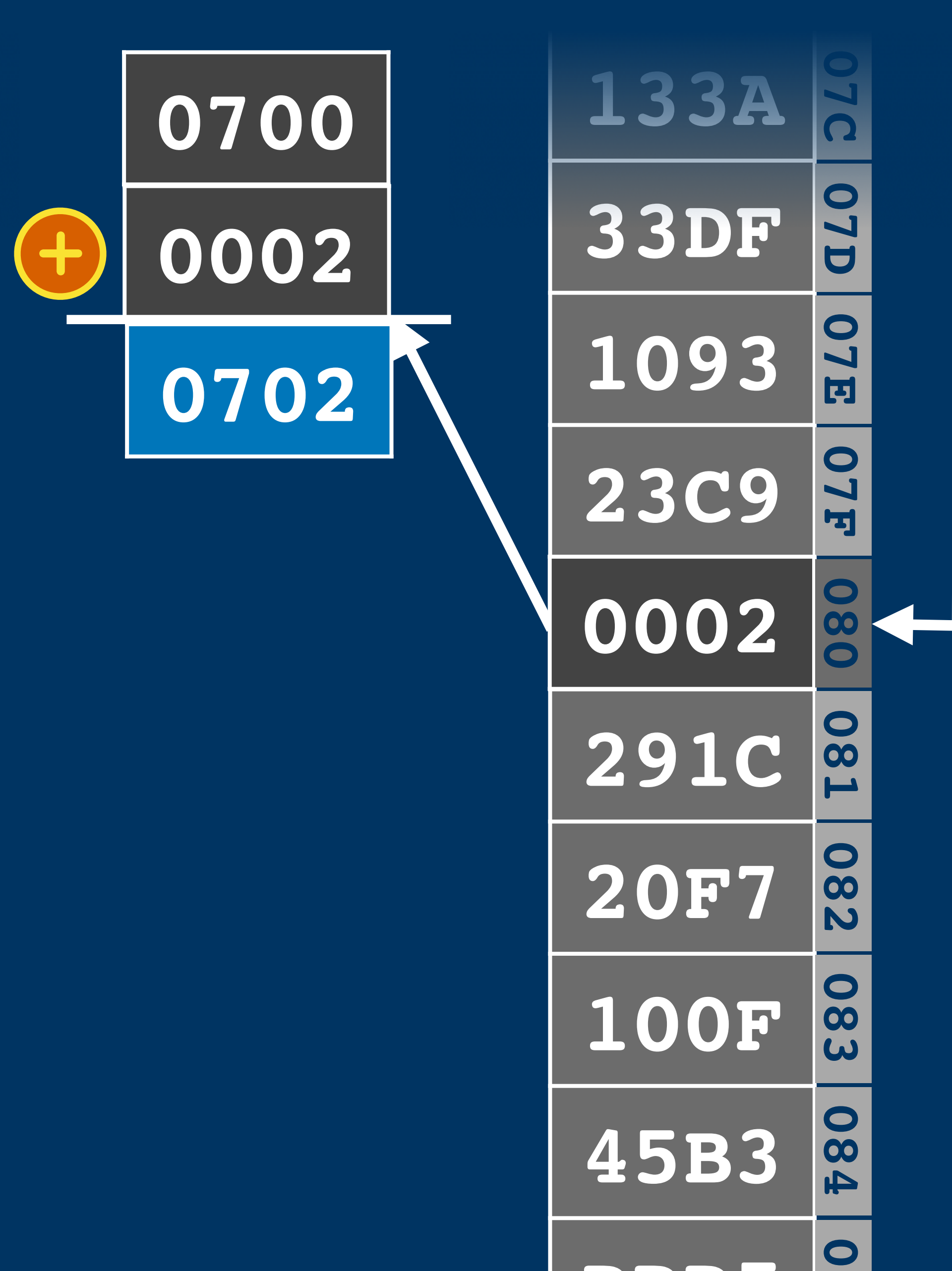
133A	07C
33DF	07D
1093	07E
23C9	07F
0002	080
291C	081
20F7	082
100F	083
45B3	084
33DF	0



## Indexing

A	0000
B	0F00
LR	05AA
EB	0000
FB	0000
PC	0421
BB	0000
0	0000

ld a, [\$700+[\$80]]



# Indexing

ld a, [\$700+[\$80]]

A	0000
B	0F00
LR	05AA
EB	0000
FB	0000
PC	0421
BB	0000
0	0000

0000	6FE
0000	6FF
0000	700
0003	701
0006	702
0009	703
000C	704
000F	705
0012	706
0015	707



0700

0002

0702

133A	07C
33DF	07D
1093	07E
23C9	07F
0002	080
291C	081
20F7	082
100F	083
45B3	084
33DF	085



Indexing

ld a, [\$700+[\$80]]

A	0006
B	0F00
LR	05AA
EB	0000
FB	0000
PC	0421
BB	0000
0	0000

0000	6FE
0000	6FF
0000	700
0003	701
0006	702
0009	703
000C	704
000F	705
0012	706
0015	707



0700
0002
0702

133A	07C
33DF	07D
1093	07E
23C9	07F
0002	080
291C	081
20F7	082
100F	083
45B3	084
3DD1	085

Indexing

ld a, [\$700+[\$80]]

A	0006
B	0F00
LR	05AA
EB	0000
FB	0000
PC	0421
BB	0000
0	0000

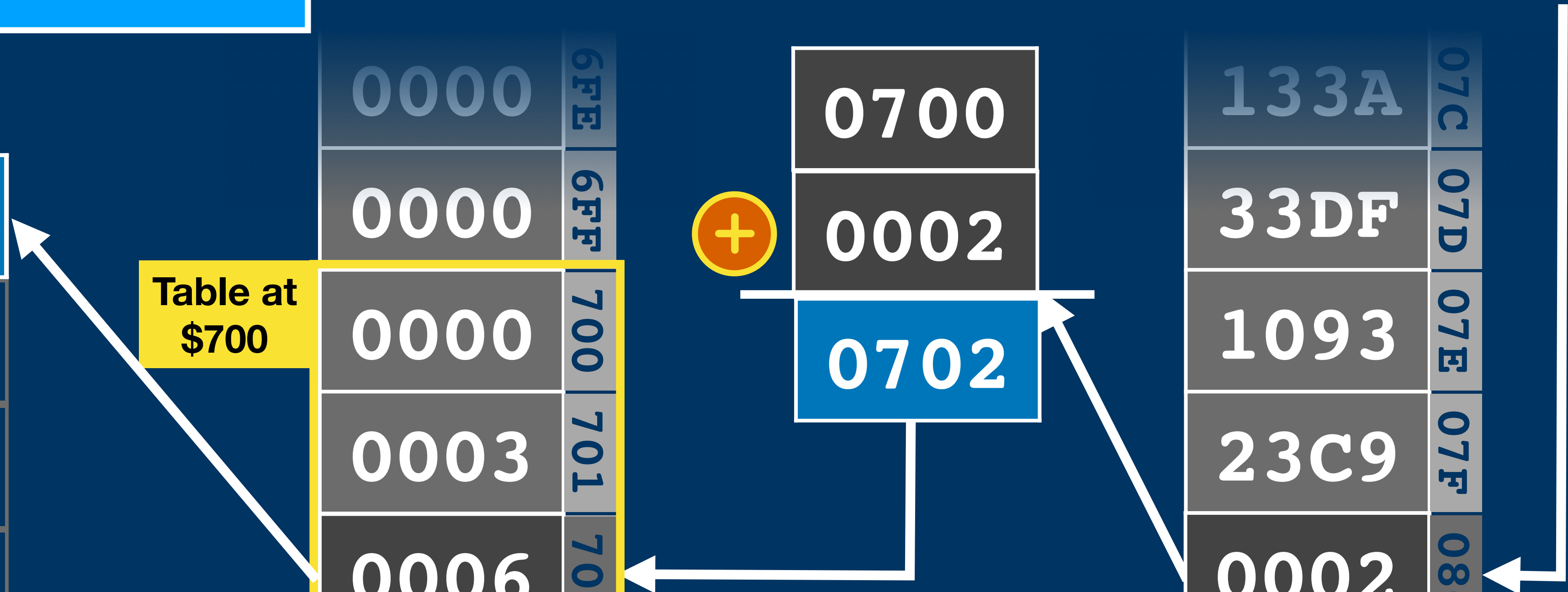
Table at \$700

0000	6FE
0000	6FF
0000	700
0003	701
0006	702
0009	703
000C	704
000F	705
0012	706
0015	707



0700
0002
0702

133A	07C
33DF	07D
1093	07E
23C9	07F
0002	080
291C	081
20F7	082
100F	083
45B3	084
33DF	085



# Indexing

ld a, [\$700+[\$80]]

A	0006
B	0F00
LR	05AA
EB	0000
FB	0000
PC	0421
BB	0000
0	0000

Table at  
\$700

0000	6FE
0000	6FF
0000	700
0003	701
0006	702
0009	703
000C	704
000F	705
0012	706
0015	707



0700
0002
0702

Index at  
\$80

133A	07C
33DF	07D
1093	07E
23C9	07F
0002	080
291C	081
20F7	082
100F	083
45B3	084
33DF	085



## Indexing

```
ld a, [[ $80 ]]
```

<b>A</b>	<b>0000</b>
<b>B</b>	<b>0F00</b>
<b>LR</b>	<b>05AA</b>
<b>EB</b>	<b>0000</b>
<b>FB</b>	<b>0000</b>
<b>PC</b>	<b>0421</b>
<b>BB</b>	<b>0000</b>
<b>0</b>	<b>0000</b>

**Indexing**

```
ld a, [0+[$80]]
```

A	0000
B	0F00
LR	05AA
EB	0000
FB	0000
PC	0421
BB	0000
0	0000

**Indexing**

```
ld a, [0+[$80]]
```

A	0000
B	0F00
LR	05AA
EB	0000
FB	0000
PC	0421
BB	0000
0	0000

133A	07C
33DF	07D
1093	07E
23C9	07F
03A0	080
291C	081
20F7	082
100F	083
45B3	084
33DF	0

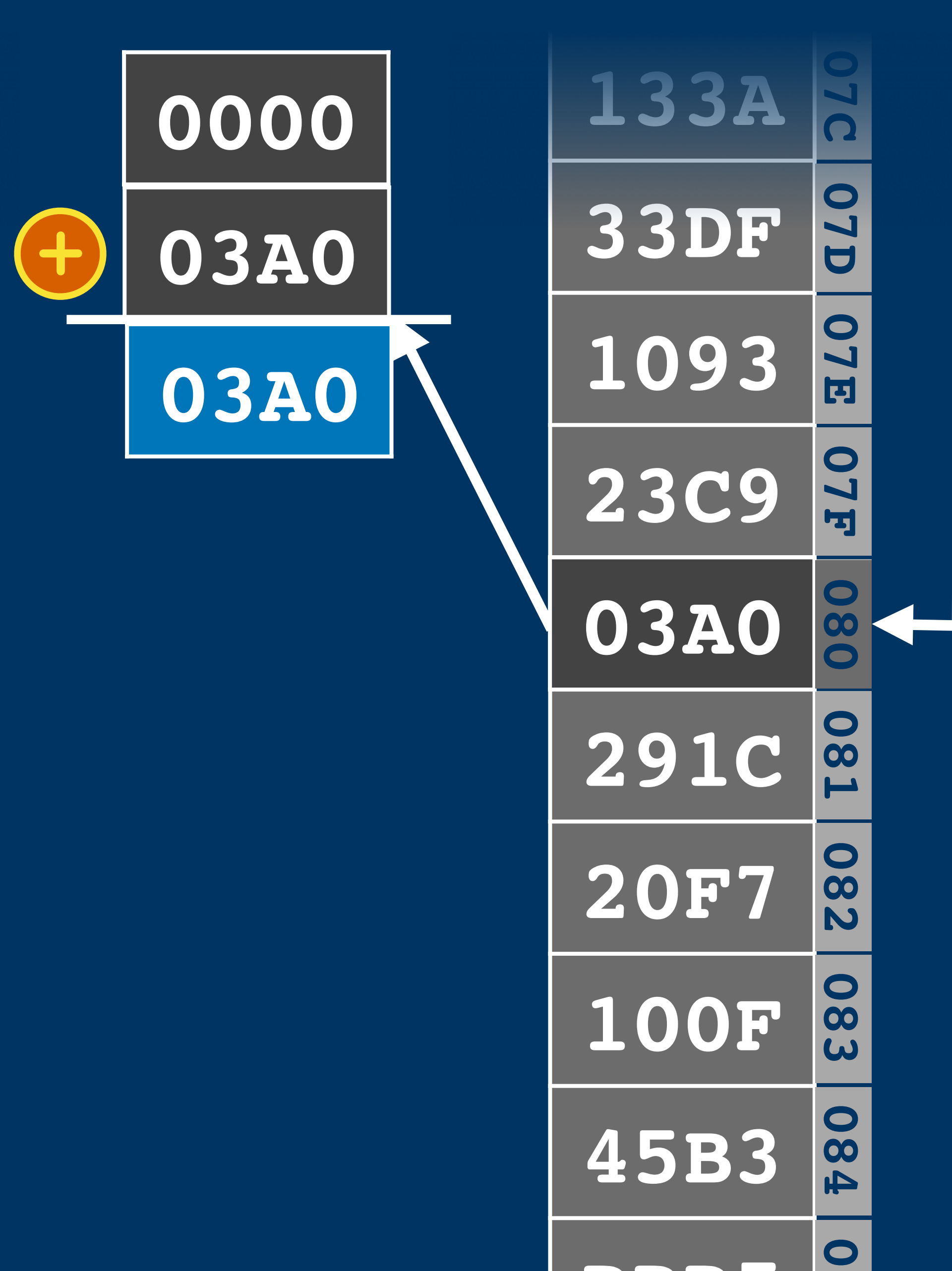




## Indexing

A	0000
B	0F00
LR	05AA
EB	0000
FB	0000
PC	0421
BB	0000
0	0000

ld a, [0+[\$80]]



Indexing

ld a, [0+[\$80]]

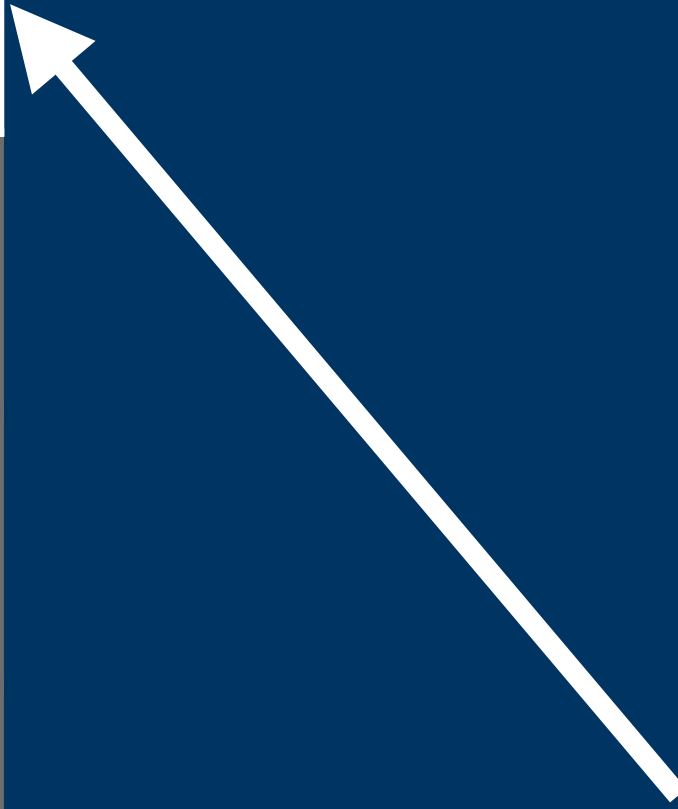
A	7700
B	0F00
LR	05AA
EB	0000
FB	0000
PC	0421
BB	0000
0	0000

0000	39C
45B3	39D
165E	39E
7120	39F
7700	3A0
3967	3A1
2B22	3A2
2F0B	3A3
49DC	3A4
2621	3A5



0000
03A0
03A0

133A	07C
33DF	07D
1093	07E
23C9	07F
03A0	080
291C	081
20F7	082
100F	083
45B3	084
33DF	0



Indexing

ld a, [0+[\$80]]

A	7700
B	0F00
LR	05AA
EB	0000
FB	0000
PC	0421
BB	0000
0	0000

0000	39C
45B3	39D
165E	39E
7120	39F
7700	3A0
3967	3A1
2B22	3A2
2F0B	3A3
49DC	3A4
262D	3A5



0000
03A0
03A0

Pointer at \$80

133A	07C
33DF	07D
1093	07E
23C9	07F
03A0	080
291C	081
20F7	082
100F	083
45B3	084
33DF	085



# Indexing

ld a, [0+[\$80]]

A	7700
B	0F00
LR	05AA
EB	0000
FB	0000
PC	0421
BB	0000
0	0000

0000	39C
45B3	39D
165E	39E
7120	39F
7700	3A0
3967	3A1
2B22	3A2
2F0B	3A3
49DC	3A4
262D	3A5



0000
03A0
03A0

133A	07C
33DF	07D
1093	07E
23C9	07F
03A0	080
291C	081
20F7	082
100F	083
45B3	084
33DF	085

Pointer destination

Pointer at \$80

Control Flow

A	0007
B	0F00
LR	05AA
EB	0000
FB	0000
PC	0400
BB	0000
0	0000

Control Flow

A	0007
B	0F00
LR	05AA
EB	0000
FB	0000
PC	0400
BB	0000
0	0000

PC →	3100	400	ld a, [\$100]
	6101	401	add a, [\$101]
	1405	402	jmp \$405
	3103	403	ld a, [\$103]
	6104	404	add a, [\$104]
	5902	405	ld [\$102], a
	0006	406	jz \$409
	140A	407	
	3103	408	ld a, [\$103]
	0006	409	jlez \$40f
	640F	40	



Control Flow

A	0007
B	0F00
LR	05AA
EB	0000
FB	0000
PC	0402
BB	0000
0	0000

PC →

3100	400	ld a, [\$100]
6101	401	add a, [\$101]
1405	402	jmp \$405
3103	403	ld a, [\$103]
6104	404	add a, [\$104]
5902	405	ld [\$102], a
0006	406	jz \$409
140A	407	
3103	408	ld a, [\$103]
0006	409	jlez \$40f
640F	40	

Control Flow

A	0007
B	0F00
LR	05AA
EB	0000
FB	0000
PC	0409
BB	0000
0	0000

PC →

3100	400	ld a, [\$100]
6101	401	add a, [\$101]
1405	402	jmp \$405
3103	403	ld a, [\$103]
6104	404	add a, [\$104]
5902	405	ld [\$102], a
0006	406	jz \$409
140A	407	
3103	408	ld a, [\$103]
0006	409	jlez \$40f
640F	40	

Jump

jmp \$62f

Jump	jmp k		TCF k	
	1000+k			1
	PC ← k			

A	0007
B	0F00
LR	05AA
EB	0000
FB	0000
PC	041F
BB	0000
0	0000



Jump

jmp \$62f

Jump	jmp k	TCF k
	1000+k	1
	PC ← k	

A	0007
B	0F00
LR	05AA
EB	0000
FB	0000
PC	041F
BB	0000
0	0000

PC →

3100	41F
162F	420
3103	421
	422
	423
	424
	425
	426
	427
	428
	429
	42A
	42B
	42C
	42D
	42E
	42F
	430
	431
	432
	433
	434
	435
	436
	437
	438
	439
	43A
	43B
	43C
	43D
	43E
	43F
	440
	441
	442
	443
	444
	445
	446
	447
	448
	449
	44A
	44B
	44C
	44D
	44E
	44F
	450
	451
	452
	453
	454
	455
	456
	457
	458
	459
	45A
	45B
	45C
	45D
	45E
	45F
	460
	461
	462
	463
	464
	465
	466
	467
	468
	469
	46A
	46B
	46C
	46D
	46E
	46F
	470
	471
	472
	473
	474
	475
	476
	477
	478
	479
	47A
	47B
	47C
	47D
	47E
	47F
	480
	481
	482
	483
	484
	485
	486
	487
	488
	489
	48A
	48B
	48C
	48D
	48E
	48F
	490
	491
	492
	493
	494
	495
	496
	497
	498
	499
	49A
	49B
	49C
	49D
	49E
	49F
	4A0
	4A1
	4A2
	4A3
	4A4
	4A5
	4A6
	4A7
	4A8
	4A9
	4AA
	4AB
	4AC
	4AD
	4AE
	4AF
	4B0
	4B1
	4B2
	4B3
	4B4
	4B5
	4B6
	4B7
	4B8
	4B9
	4BA
	4BB
	4BC
	4BD
	4BE
	4BF
	4C0
	4C1
	4C2
	4C3
	4C4
	4C5
	4C6
	4C7
	4C8
	4C9
	4CA
	4CB
	4CC
	4CD
	4CE
	4CF
	4D0
	4D1
	4D2
	4D3
	4D4
	4D5
	4D6
	4D7
	4D8
	4D9
	4DA
	4DB
	4DC
	4DD
	4DE
	4DF
	4E0
	4E1
	4E2
	4E3
	4E4
	4E5
	4E6
	4E7
	4E8
	4E9
	4EA
	4EB
	4EC
	4ED
	4EE
	4EF
	4F0
	4F1
	4F2
	4F3
	4F4
	4F5
	4F6
	4F7
	4F8
	4F9
	4FA
	4FB
	4FC
	4FD
	4FE
	4FF
	500
	501
	502
	503
	504
	505
	506
	507
	508
	509
	50A
	50B
	50C
	50D
	50E
	50F
	510
	511
	512
	513
	514
	515
	516
	517
	518
	519
	51A
	51B
	51C
	51D
	51E
	51F
	520
	521
	522
	523
	524
	525
	526
	527
	528
	529
	52A
	52B
	52C
	52D
	52E
	52F
	530
	531
	532
	533
	534
	535
	536
	537
	538
	539
	53A
	53B
	53C
	53D
	53E
	53F
	540
	541
	542
	543
	544
	545
	546
	547
	548
	549
	54A
	54B
	54C
	54D
	54E
	54F
	550
	551
	552
	553
	554
	555
	556
	557
	558
	559
	55A
	55B
	55C
	55D
	55E
	55F
	560
	561
	562
	563
	564
	565
	566
	567
	568
	569
	56A
	56B
	56C
	56D
	56E
	56F
	570
	571
	572
	573
	574
	575
	576
	577
	578
	579
	57A
	57B
	57C
	57D
	57E
	57F
	580
	581
	582
	583
	584
	585
	586
	587
	588
	589
	58A
	58B
	58C
	58D
	58E
	58F
	590
	591
	592
	593
	594
	595
	596
	597
	598
	599
	59A
	59B
	59C
	59D
	59E
	59F
	5A0
	5A1
	5A2
	5A3
	5A4
	5A5
	5A6
	5A7
	5A8
	5A9
	5AA
	5AB
	5AC
	5AD
	5AE
	5AF
	5B0
	5B1
	5B2
	5B3
	5B4
	5B5
	5B6
	5B7
	5B8
	5B9
	5BA
	5BB
	5BC
	5BD
	5BE
	5BF
	5C0
	5C1
	5C2
	5C3
	5C4
	5C5
	5C6
	5C7
	5C8
	5C9
	5CA
	5CB
	5CC
	5CD
	5CE
	5CF
	5D0
	5D1
	5D2
	5D3
	5D4
	5D5
	5D6
	5D7
	5D8
	5D9
	5DA
	5DB
	5DC
	5DD
	5DE
	5DF
	5E0
	5E1
	5E2
	5E3
	5E4
	5E5
	5E6
	5E7
	5E8
	5E9
	5EA
	5EB
	5EC
	5ED
	5EE
	5EF
	5F0
	5F1
	5F2
	5F3
	5F4
	5F5
	5F6
	5F7
	5F8
	5F9
	5FA
	5FB
	5FC
	5FD
	5FE
	5FF
	600
	601
	602
	603
	604
	605
	606
	607
	608
	609
	60A
	60B
	60C
	60D
	60E
	60F
	610
	611
	612
	613
	614
	615
	616
	617
	618
	619
	61A
	61B
	61C
	61D
	61E
	61F
	620
	621
	622
	623
	624
	625
	626
	627
	628
	629
	62A
	62B
	62C
	62D
	62E
	62F
	630
	631
	632
	633
	634
	635
	636
	637
	638
	639
	63A
	63B
	63C
	63D
	63E
	63F
	640
	641
	642
	643
	644
	645
	646
	647
	648
	649
	64A
	64B
	64C
	64D
	64E
	64F
	650
	651
	652
	653
	654
	655
	656
	657
	658
	659
	65A
	65B
	65C
	65D
	65E
	65F
	660
	661
	662
	663
	664
	665
	666
	667
	668
	669
	66A
	66B
	66C
	66D
	66E
	66F
	670
	671
	672
	673
	674
	675
	676
	677
	678
	679
	67A
	67B
	67C
	67D
	67E
	67F
	680
	681
	682
	683
	684
	685
	686
	687
	688
	689
	68A
	68B
	68C
	68D
	68E
	68F
	690
	691
	692
	693
	694
	695
	696
	697
	698
	699
	69A
	69B
	69C
	69D
	69E
	69F
	6A0
	6A1
	6A2
	6A3
	6A4
	6A5
	6A6
	6A7
	6A8
	6A9
	6AA
	6AB
	6AC
	6AD
	6AE
	6AF
	6B0
	6B1
	6B2
	6B3
	6B4
	6B5
	6B6
	6B7
	6B8
	6B9
	6BA
	6BB
	6BC
	6BD
	6BE
	6BF
	6C0
	6C1
	6C2
	6C3
	6C4
	6C5
	6C6
	6C7
	6C8
	6C9
	6CA
	6CB
	6CC
	6CD
	6CE
	6CF
	6D0
	6D1
	6D2
	6D3
	6D4
	6D5
	6D6
	6D7
	6D8
	6D9
	6DA
	6DB
	6DC
	6DD
	6DE
	6DF
	6E0
	6E1
	6E2
	6E3
	6E4
	6E5
	6E6
	6E7
	6E8
	6E9
	6EA
	6EB
	6EC
	6ED
	6EE
	6EF
	6F0
	6F1
	6F2
	6F3
	6F4
	6F5
	6F6
	6F7
	6F8
	6F9
	6FA
	6FB
	6FC
	6FD
	6FE
	6FF
	700
	701
	702
	703
	704
	705
	706
	707
	708
	709
	70A
	70B
	70C
	70D
	70E

Jump

jmp \$62f

Jump

jmp k

TCF k

1000+k

1

PC ← k

A 0007

B 0F00

LR 05AA

EB 0000

FB 0000

PC 0420

BB 0000

0 0000

PC →

3100

41F

ld a, [\$100]

162F

420

jmp \$62f

3103

421

ld a, [\$103]

5880

62F

ld [\$80], a

3102

630

ld a, [\$102]

Jump

jmp \$62f

Jump

jmp k

TCF k

1000+k

1

PC ← k

A 0007

B 0F00

LR 05AA

EB 0000

FB 0000

PC 062F

BB 0000

0 0000

3100

41F

ld a, [\$100]

162F

420

jmp \$62f

3103

421

ld a, [\$103]

5880

62F

PC →

ld [\$80], a

3102

630

ld a, [\$102]



# Conditional Jumps

Jump	jmp k		TCF k	
	1000+k			1
	PC ← k			

# Conditional Jumps

Jump	jmp k	TCF k
	1000+k	1
	PC ← k	

Jump = 0	jz k	BZF k
	0006, 1000+k	2-3
	if A=±0: PC ← k	

Jump ≤ 0	jlez k	BZMF k
	0006, 6000+k	2-3
	if A≤±0: PC ← k	

# Conditional Jumps

Jump	jmp k	TCF k
	1000+k	1
	PC ← k	
Jump = 0	jz k	BZF k
	0006, 1000+k	2-3
	if A=±0: PC ← k	
Jump ≤ 0	jlez k	BZMF k
	0006, 6000+k	2-3
	if A≤±0: PC ← k	



Conditional Jumps

Jump	jmp k	TCF k
	1000+k	1
	PC ← k	
Jump = 0	jz k	BZF k
	0006, 1000+k	2-3
	if A=±0: PC ← k	
Jump ≤ 0	jlez k	BZMF k
	0006, 6000+k	2-3
	if A≤±0: PC ← k	

Count, Compare and Skip	ccs [k]	CCS k
	1000+k	2
	<pre>A ← DABS(M[k]) if M[k]&gt;+0:     PC ← PC if M[k]=+0:     PC ← PC+1 if M[k]&lt;-0:     PC ← PC+2 if M[k]=-0:     PC ← PC+3</pre>	

# Conditional Jumps

Count, Compare and Skip	ccs [k]	CCS k
	1000+k	2
	<div>A ← DABS(M[k]) if M[k]&gt;+0:     PC ← PC if M[k]=+0:     PC ← PC+1 if M[k]&lt;-0:     PC ← PC+2 if M[k]=-0:     PC ← PC+3</div>	

# Conditional Jumps

Count, Compare and Skip	ccs [k]	CCS k
	1000+k	2
	<div>A ← DABS(M[k]) if M[k]&gt;+0:     PC ← PC if M[k]=+0:     PC ← PC+1 if M[k]&lt;-0:     PC ← PC+2 if M[k]=-0:     PC ← PC+3</div>	

			ccs [\$200]
	1200	41F	
1	1430	420	jmp \$430
2	1440	421	jmp \$440
3	1450	422	jmp \$450
4	3080	62E	ld a, [\$80]



Conditional Jumps

Count, Compare and Skip	ccs [k]	CCS k
	1000+k	2
	<div>A ← DABS(M[k]) if M[k]&gt;+0:     PC ← PC if M[k]=+0:     PC ← PC+1 if M[k]&lt;-0:     PC ← PC+2 if M[k]=-0:     PC ← PC+3</div>	

575D	E
1200	41F
1430	420
3080	421
5881	422
2882	62E
6EAE	62F

```
ccs [$200]
jmp $430
ld a, [$80]
ld [$81], a
inc [$82]
```

Conditional Jumps

Count, Compare and Skip	ccs [k]	CCS k
	1000+k	2
	<div>A ← DABS(M[k]) if M[k]&gt;+0:     PC ← PC if M[k]=+0:     PC ← PC+1 if M[k]&lt;-0:     PC ← PC+2 if M[k]=-0:     PC ← PC+3</div>	

	1200	41F	ccs [\$200]	
1	1430	420	jmp \$430	>+0
2	3080	421	ld a, [\$80]	=+0
	5881	422	ld [\$81], a	
	2882	62E	inc [\$82]	

Conditional Jumps

Count, Compare and Skip	ccs [k]	CCS k
	1000+k	2
	<div>A ← DABS(M[k]) if M[k]&gt;+0:     PC ← PC if M[k]=+0:     PC ← PC+1 if M[k]&lt;-0:     PC ← PC+2 if M[k]=-0:     PC ← PC+3</div>	

575D	E
1200	41F
1F00	420
1F00	421
1430	422
3080	62E
6EAF	62F

ccs [\$200]

jmp \$f00

jmp \$f00

jmp \$430

ld a, [\$80]



Conditional Jumps

Count, Compare and Skip	ccs [k]	CCS k
	1000+k	2
	<div>A ← DABS(M[k]) if M[k]&gt;+0:     PC ← PC if M[k]=+0:     PC ← PC+1 if M[k]&lt;-0:     PC ← PC+2 if M[k]=-0:     PC ← PC+3</div>	

	1200	41F	ccs [\$200]
	1F00	420	jmp \$f00
	1F00	421	jmp \$f00
3	1430	422	jmp \$430
4	3080	62E	ld a, [\$80]

>-0

=-0

Conditional Jumps

Count, Compare and Skip	ccs [k]	CCS k
	1000+k	2
	<div>A ← DABS(M[k]) if M[k]&gt;+0:     PC ← PC if M[k]=+0:     PC ← PC+1 if M[k]&lt;-0:     PC ← PC+2 if M[k]=-0:     PC ← PC+3</div>	

	1200	41F	ccs [\$200]
	1F00	420	jmp <del>\$f00</del> error
	1F00	421	jmp <del>\$f00</del> error
3	1430	422	jmp \$430
4	3080	62E	ld a, [\$80]

>-0

=-0

Conditional Jumps

Count, Compare and Skip	ccs [k]	CCS k
	1000+k	2
	<div>A ← DABS(M[k]) if M[k]&gt;+0:     PC ← PC if M[k]=+0:     PC ← PC+1 if M[k]&lt;-0:     PC ← PC+2 if M[k]=-0:     PC ← PC+3</div>	

375D	E
1200	41F
1430	420
3080	421
5881	422
2882	62


```
ccs a
jmp $410
ld a, [$80]
ld [$81], a
```



Conditional Jumps

Count, Compare and Skip	ccs [k]	CCS k
	1000+k	2
	<div>A ← DABS(M[k]) if M[k]&gt;+0:     PC ← PC if M[k]=+0:     PC ← PC+1 if M[k]&lt;-0:     PC ← PC+2 if M[k]=-0:     PC ← PC+3</div>	

	1200	41F	ccs a	
1	1430	420	jmp \$410	>+0
2	3080	421	ld a, [\$80]	=+0
	5881	422	ld [\$81], a	



Call

call \$e80

Call	call k		TC k	
	0000+k			1
	LR ← PC   PC ← k			

A	0007
B	0F00
LR	05AA
EB	0000
FB	0000
PC	041F
BB	0000
0	0000

Call

call \$e80

Call

call k

TC k

0000+k

1

LR ← PC | PC ← k

A 0007

B 0F00

LR 05AA

EB 0000

FB 0000

PC 041F

BB 0000

0 0000

PC →

3100

41F

ld a, [\$100]

0E80

420

call \$e80

3103

421

ld a, [\$103]

5880

E80

ld [\$80], a

0002

E81

ret



Call

call \$e80

call

call k

TC k

0000+k

1

LR ← PC | PC ← k

A 0007

B 0F00

LR 05AA

EB 0000

FB 0000

PC 0420

BB 0000

0 0000

PC →

3100

41F

ld a, [\$100]

0E80

420

call \$e80

3103

421

ld a, [\$103]

5880

E80

ld [\$80], a

0002

E81

ret

Call

call \$e80

call

call k

TC k

0000+k

1

LR ← PC | PC ← k

A 0007

B 0F00

LR 05AA

EB 0000

FB 0000

PC 0421

BB 0000

0 0000

PC →

3100

41F

ld a, [\$100]

0E80

420

call \$e80

3103

421

ld a, [\$103]

5880

E80

ld [\$80], a

0002

E81

ret

Call

call \$e80

call

call k

TC k

0000+k

1

LR ← PC | PC ← k

A 0007

B 0F00

LR 0421

EB 0000

FB 0000

PC 0421

BB 0000

0 0000

PC →

3100

41F

ld a, [\$100]

0E80

420

call \$e80

3103

421

ld a, [\$103]

5880

E80

ld [\$80], a

0002

E81

ret



Call

call \$e80

call

call k

TC k

0000+k

1

LR ← PC | PC ← k

A 0007

B 0F00

LR 0421

EB 0000

FB 0000

PC 0E80

BB 0000

0 0000

PC →

3100

41F

ld a, [\$100]

0E80

420

call \$e80

3103

421

ld a, [\$103]

5880

E80

ld [\$80], a

0002

E81

ret

Call

call \$e80

call

call k

TC k

0000+k

1

LR ← PC | PC ← k

A 0007

B 0F00

LR 0421

EB 0000

FB 0000

PC 0E80

BB 0000

0 0000

3100

41F

ld a, [\$100]

0E80

420

call \$e80

3103

421

ld a, [\$103]

5880

E80

ld [\$80], a

0002

E81

ret

PC →

Return

ret

Return

ret

RETURN

0002

alias

1

LR ← PC | PC ← 2

A 0007

B 0F00

LR 0421

EB 0000

FB 0000

PC 0E80

BB 0000

0 0000

3100

0E80

3103

5880

0002

ld a, [\$100]

call \$e80

ld a, [\$103]

ld [\$80], a

ret

PC →



Return

ret

Return

ret

RETURN

0002

alias

1

LR ← PC | PC ← 2

A 0007

B 0F00

LR 0421

EB 0000

FB 0000

PC 0E81

BB 0000

0 0000

3100

41F

ld a, [\$100]

0E80

420

call \$e80

3103

421

ld a, [\$103]

5880

E80

ld [\$80], a

0002

E81

ret

PC →

Return

ret

Return

ret

RETURN

0002

alias

1

LR ← PC | PC ← 2

A 0007

B 0F00

LR 0421

EB 0000

FB 0000

PC 0421

BB 0000

0 0000

3100

41F

ld a, [\$100]

0E80

420

call \$e80

3103

421

ld a, [\$103]

5880

E80

ld [\$80], a

0002

E81

ret

PC →

Return

ret

Return

ret

RETURN

0002

alias

1

LR ← PC | PC ← 2

A 0007

B 0F00

LR 0421

EB 0000

FB 0000

PC 0421

BB 0000

0 0000

PC →

3100

41F

ld a, [\$100]

0E80

420

call \$e80

3103

421

ld a, [\$103]

5880

E80

ld [\$80], a

0002

E81

ret



Exchange LR

Exchg LR	xchg lr, [k]		QXCH k	
	0006, 2400+k			3
	LR ↔ M[k]			

Exchange LR

Exchg LR	xchg lr, [k]		QXCH k	
	0006, 2400+k			3
	LR ↔ M[k]			

call \$ee0

ret

Exchange LR

Exchg LR	xchg lr, [k]		QXCH k	
	0006, 2400+k			3
	LR ↔ M[k]			

```
xchg lr, [$4ff]
call $ee0
xchg lr, [$4ff]
ret
```



# Registers

A

B

LR

EB

FB

PC

BB

0

## Registers

**A**

**0007**

**B**

**LR**

**EB**

**FB**

**PC**

**BB**

**0**

## Registers

**A** 0007

**B** 0F00

LR

EB

FB

PC

BB

0



## Registers

**A** 0007

**B** 0F00

LR

EB

FB

**PC** 0421

BB

0

## Registers

**A** 0007

**B** 0F00

**LR** 05AA

**EB**

**FB**

**PC** 0421

**BB**

**0**

## Registers

**A** 0007

**B** 0F00

**LR** 05AA

**EB**

**FB**

**PC** 0421

**BB**

**0** 0000



## Registers

**A**    **0007**

**B**    **0F00**

**LR**   **05AA**

**EB**   **0000**

**FB**   **0000**

**PC**   **0421**

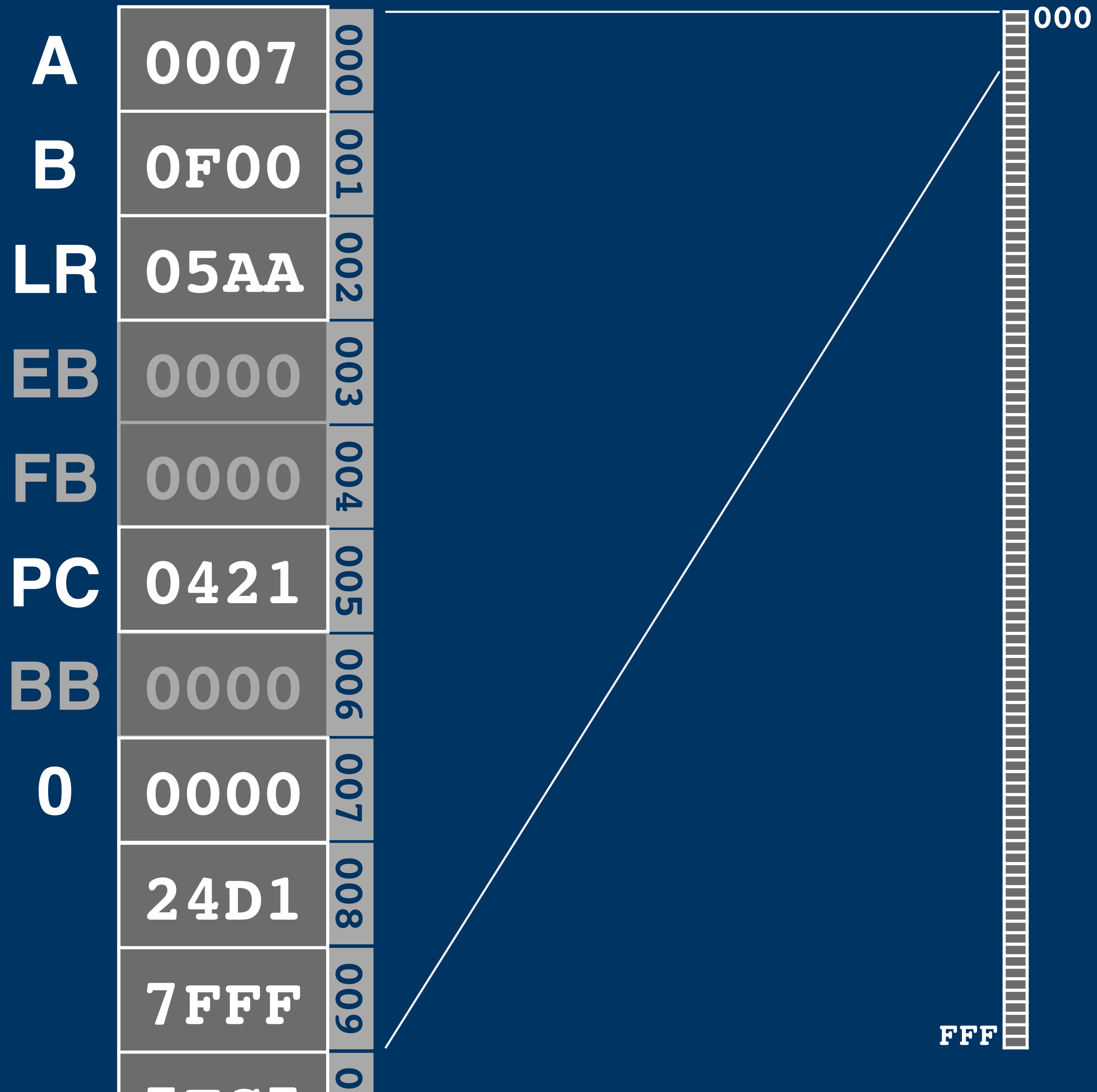
**BB**   **0000**

**0**    **0000**

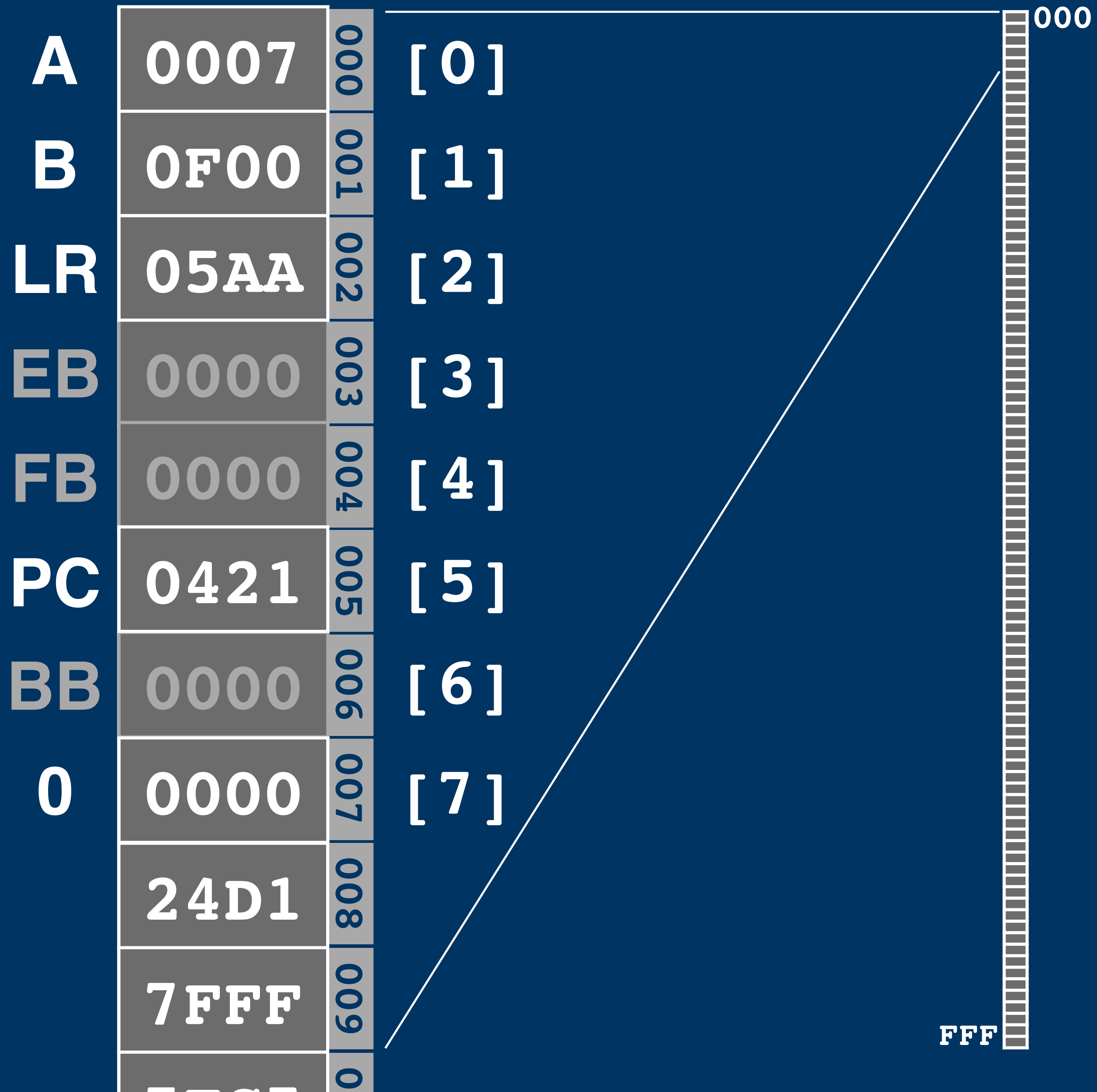
## Registers

<b>A</b>	<b>0007</b>	<b>000</b>
<b>B</b>	<b>0F00</b>	<b>001</b>
<b>LR</b>	<b>05AA</b>	<b>002</b>
<b>EB</b>	<b>0000</b>	<b>003</b>
<b>FB</b>	<b>0000</b>	<b>004</b>
<b>PC</b>	<b>0421</b>	<b>005</b>
<b>BB</b>	<b>0000</b>	<b>006</b>
<b>0</b>	<b>0000</b>	<b>007</b>

<b>A</b>	<b>0007</b>	<b>000</b>
<b>B</b>	<b>0F00</b>	<b>001</b>
<b>LR</b>	<b>05AA</b>	<b>002</b>
<b>EB</b>	<b>0000</b>	<b>003</b>
<b>FB</b>	<b>0000</b>	<b>004</b>
<b>PC</b>	<b>0421</b>	<b>005</b>
<b>BB</b>	<b>0000</b>	<b>006</b>
<b>0</b>	<b>0000</b>	<b>007</b>







Memory-Mapped Registers

A	0007	000	[0]
B	0F00	001	[1]
LR	05AA	002	[2]
EB	0000	003	[3]
FB	0000	004	[4]
PC	0421	005	[5]
BB	0000	006	[6]
0	0000	007	[7]

# Memory-Mapped Registers

A	0007	000	[0]
B	0F00	001	[1]
LR	05AA	002	[2]
EB	0000	003	[3]
FB	0000	004	[4]
PC	0421	005	[5]
BB	0000	006	[6]
0	0000	007	[7]

ld a, b

Memory-Mapped Registers

A	0007	000	[0]
B	0F00	001	[1]
LR	05AA	002	[2]
EB	0000	003	[3]
FB	0000	004	[4]
PC	0421	005	[5]
BB	0000	006	[6]
0	0000	007	[7]

ld a, b                   ≡       ld a, [1]



Memory-Mapped Registers

A	0007	000	[0]	ld a, b	≡	ld a, [1]
B	0F00	001	[1]			
LR	05AA	002	[2]	ld a, #0		
EB	0000	003	[3]			
FB	0000	004	[4]			
PC	0421	005	[5]			
BB	0000	006	[6]			
0	0000	007	[7]			

Memory-Mapped Registers

A	0007	000	[0]	ld a, b	≡	ld a, [1]
B	0F00	001	[1]			
LR	05AA	002	[2]	ld a, #0	≡	ld a, [7]
EB	0000	003	[3]			
FB	0000	004	[4]			
PC	0421	005	[5]			
BB	0000	006	[6]			
0	0000	007	[7]			

Memory-Mapped Registers

A	0007	000	[0]	ld a, b	≡	ld a, [1]
B	0F00	001	[1]			
LR	05AA	002	[2]	ld a, #0	≡	ld a, [7]
EB	0000	003	[3]			
FB	0000	004	[4]	inc a		
PC	0421	005	[5]			
BB	0000	006	[6]			
0	0000	007	[7]			

# Memory-Mapped Registers

A	0007	000	[0]	ld a, b	≡	ld a, [1]
B	0F00	001	[1]			
LR	05AA	002	[2]	ld a, #0	≡	ld a, [7]
EB	0000	003	[3]			
FB	0000	004	[4]	inc a	≡	inc [0]
PC	0421	005	[5]			
BB	0000	006	[6]			
0	0000	007	[7]			



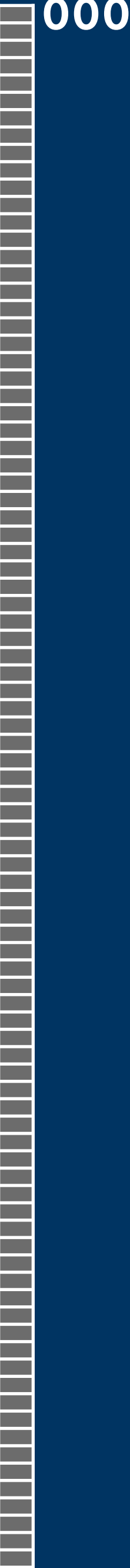
Memory-Mapped Registers

A	0007	000	[0]	ld a, b	≡	ld a, [1]
B	0F00	001	[1]			
LR	05AA	002	[2]	ld a, #0	≡	ld a, [7]
EB	0000	003	[3]			
FB	0000	004	[4]	inc a	≡	inc [0]
PC	0421	005	[5]			
BB	0000	006	[6]	ld a, [b]		
0	0000	007	[7]			

# Memory-Mapped Registers

A	0007	000	[0]	ld a, b	≡	ld a, [1]
B	0F00	001	[1]			
LR	05AA	002	[2]	ld a, #0	≡	ld a, [7]
EB	0000	003	[3]			
FB	0000	004	[4]	inc a	≡	inc [0]
PC	0421	005	[5]			
BB	0000	006	[6]	ld a, [b]	≡	ld a, [[1]]
0	0000	007	[7]			

# Memory Map



Memory Map

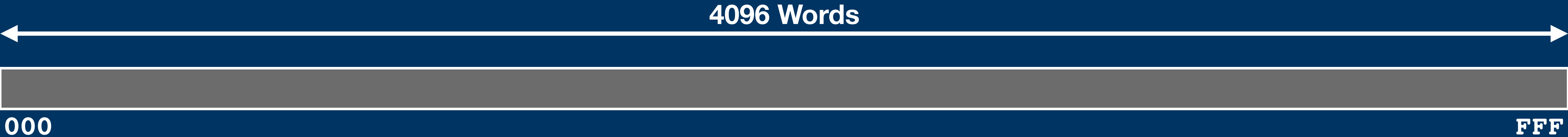




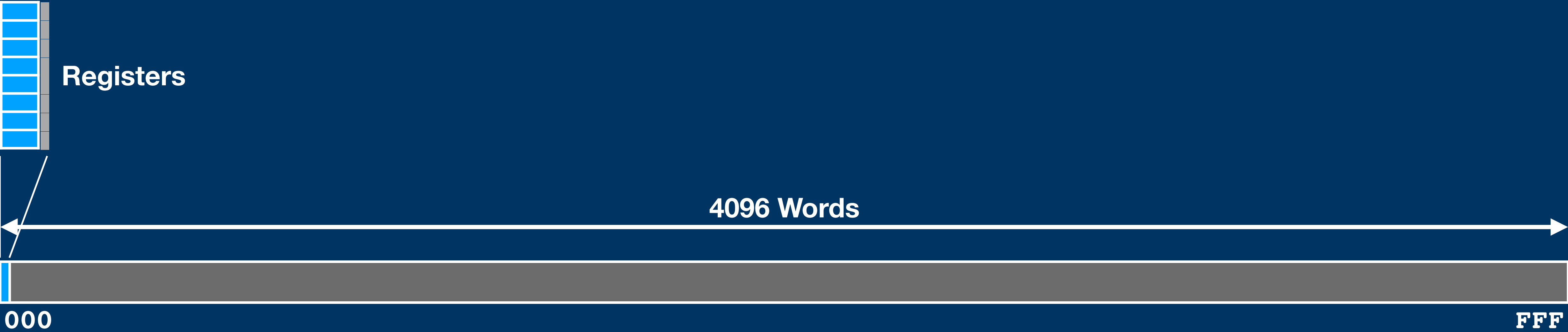
Memory Map



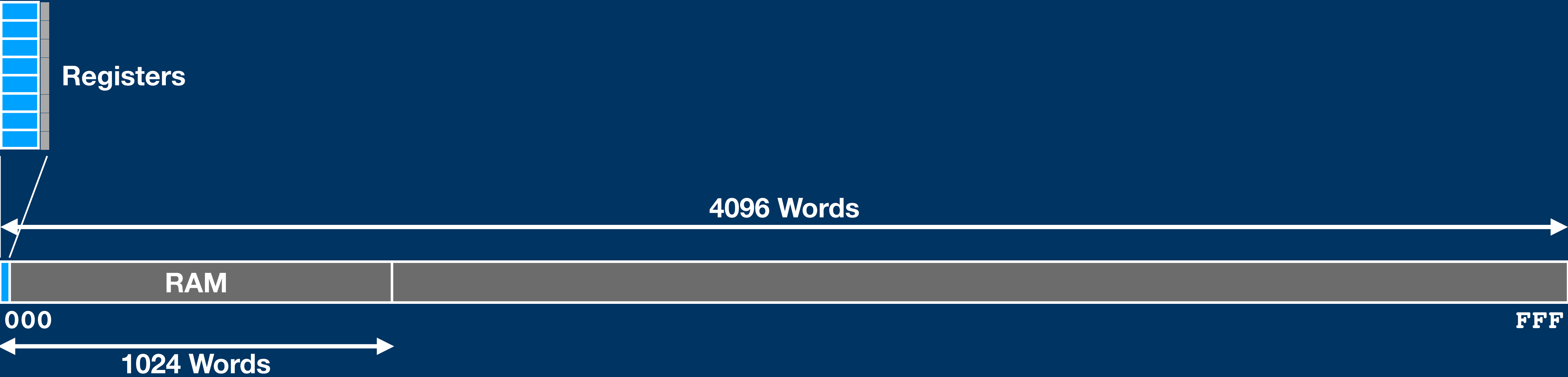
Memory Map



# Memory Map

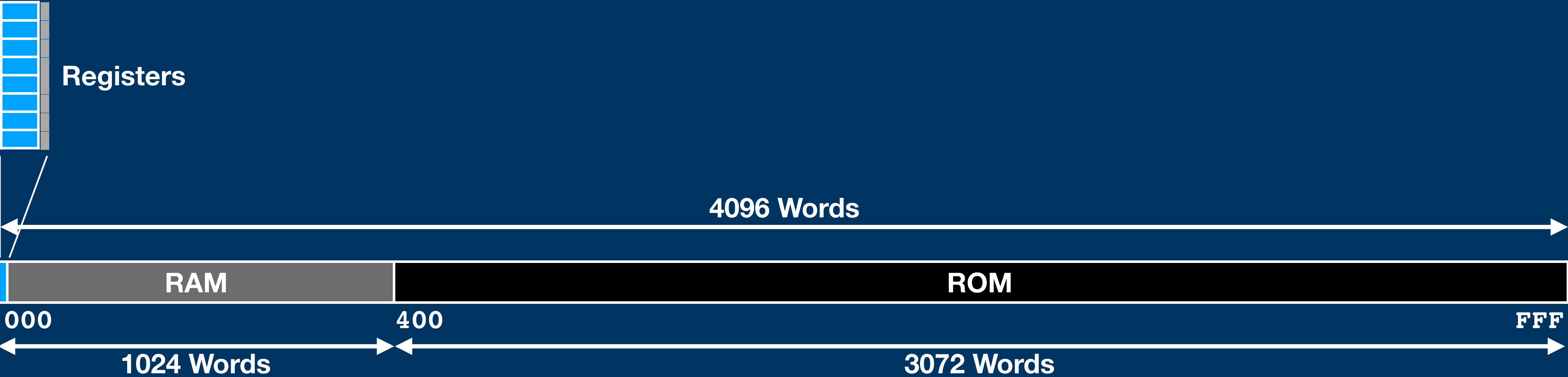


# Memory Map





# Memory Map



Memory Map: RAM

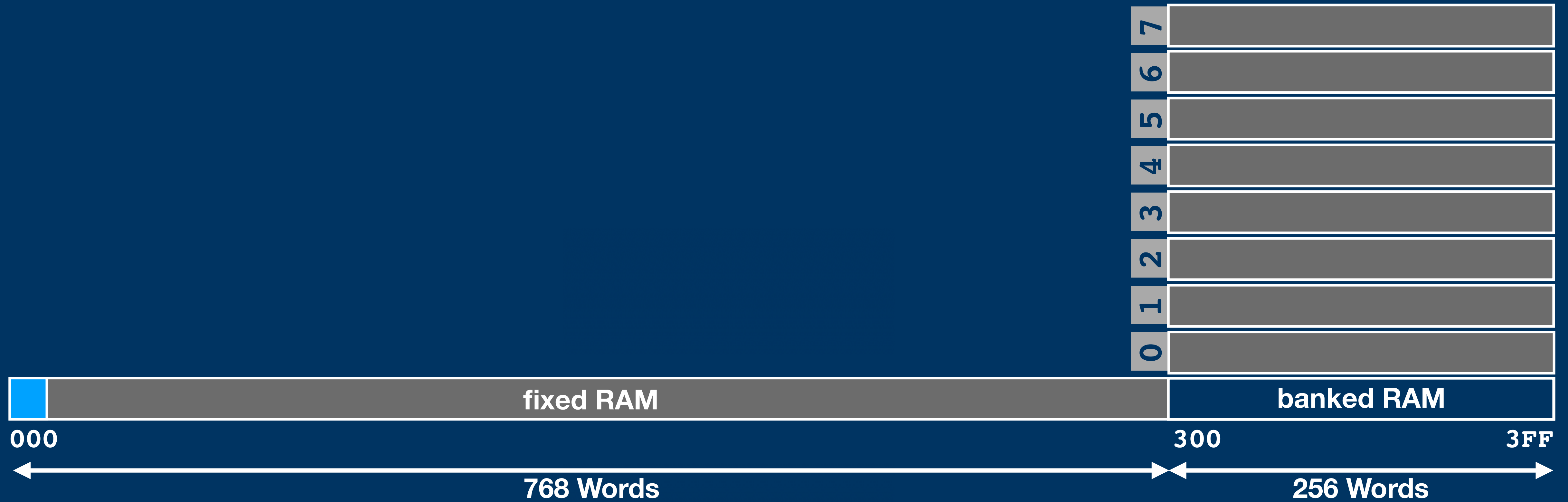


000

RAM

3FF

# Memory Map: RAM



# Memory Map: RAM

A	0007
B	0F00
LR	0421
<b>EB</b>	<b>0000</b>
FB	0000
DC	0F80

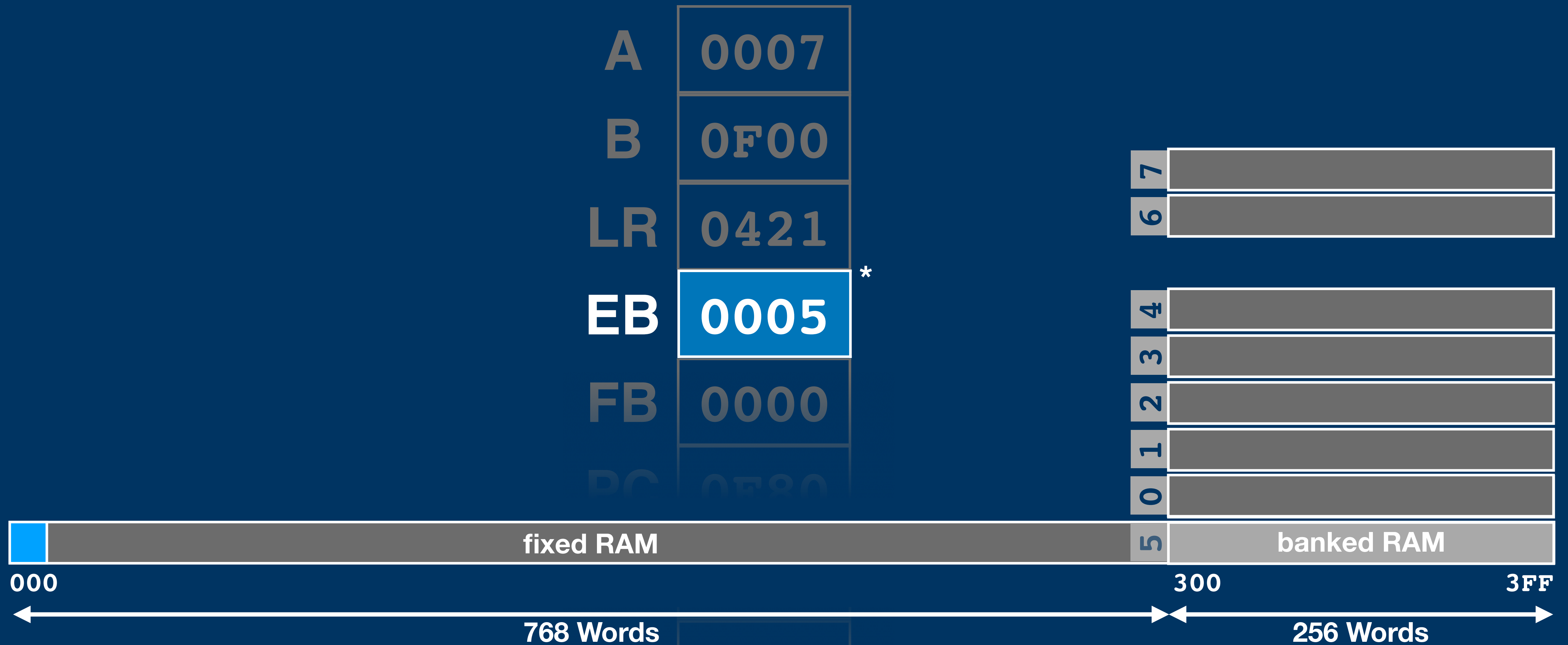
7	
6	
5	
4	
3	
2	
1	



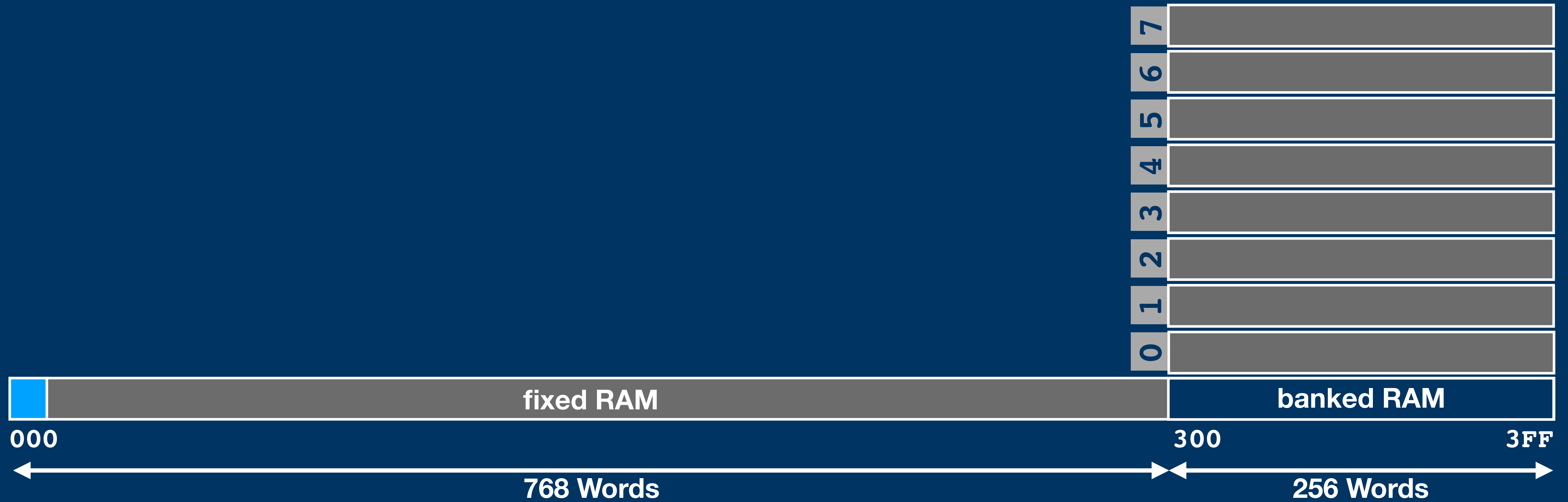


# Memory Map: RAM

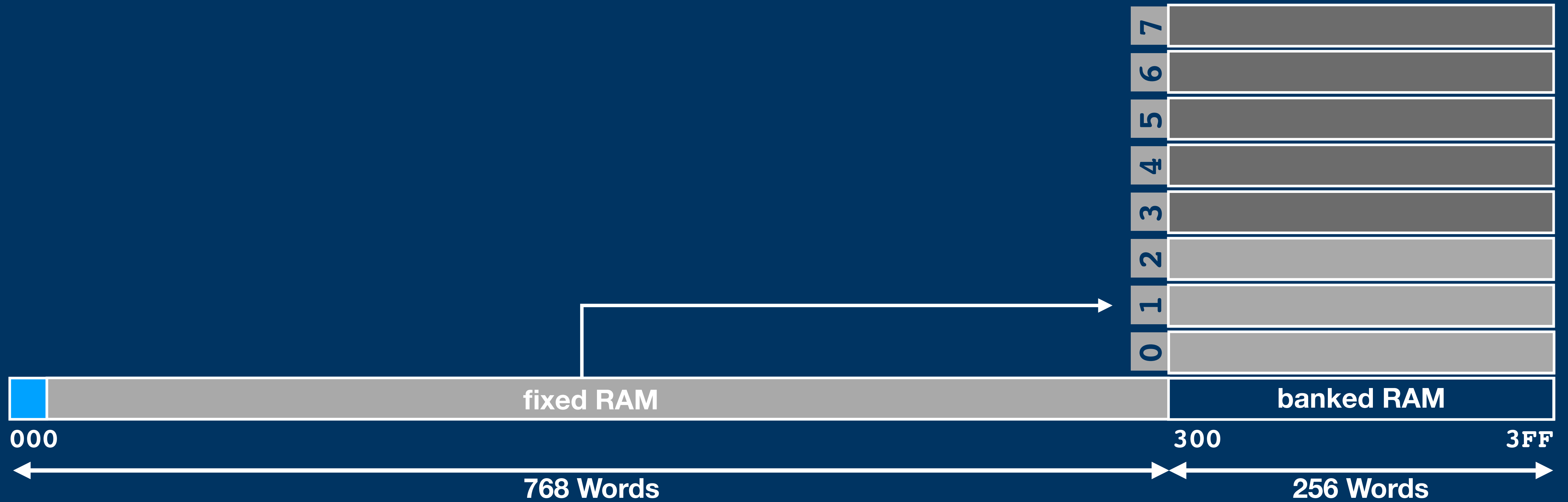
\* the 3 bits are actually << 8,  
so EB will contain 0500



# Memory Map: RAM



# Memory Map: RAM



Memory Map: RAM



000

RAM

3FF



# Memory Map

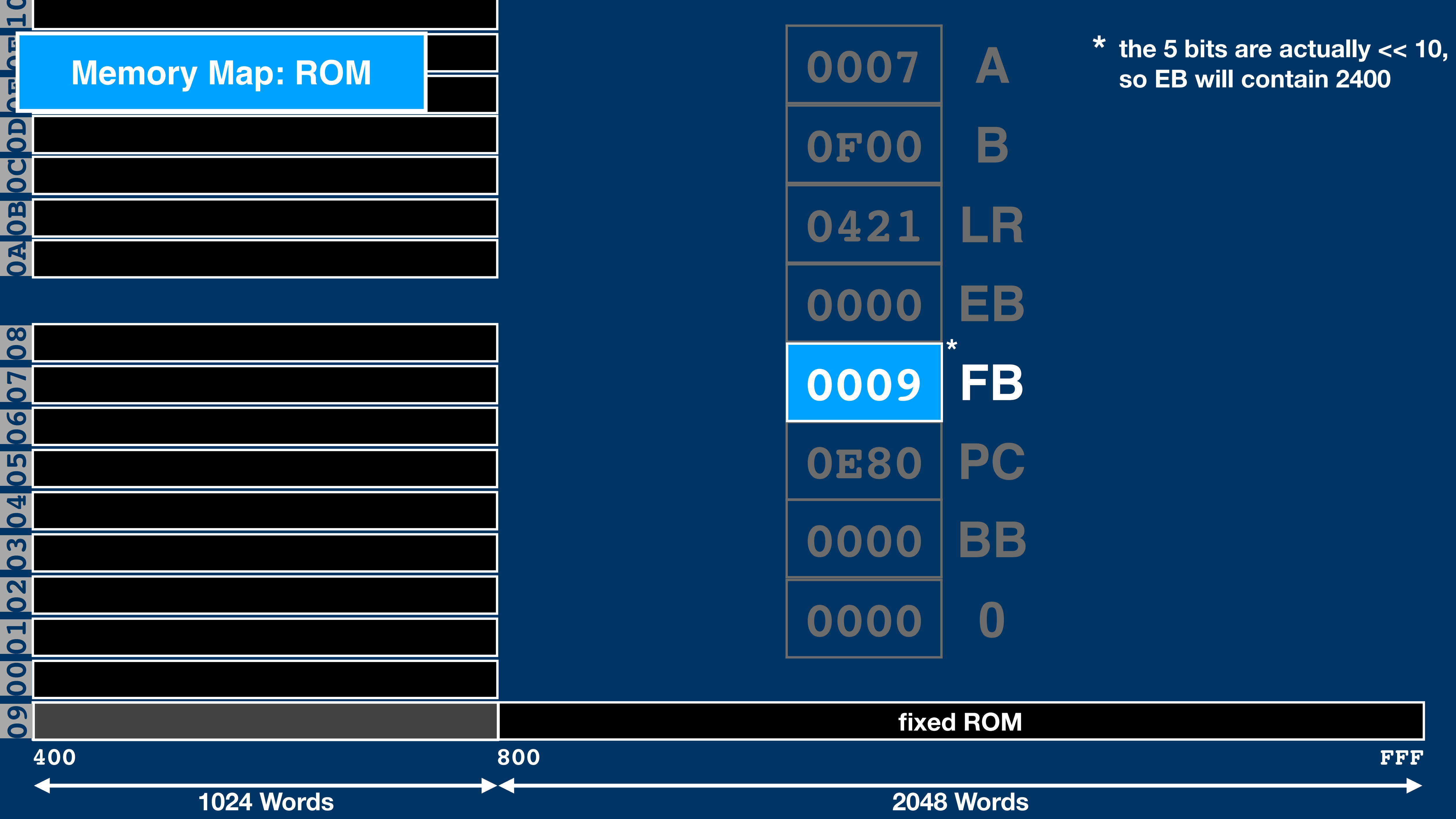


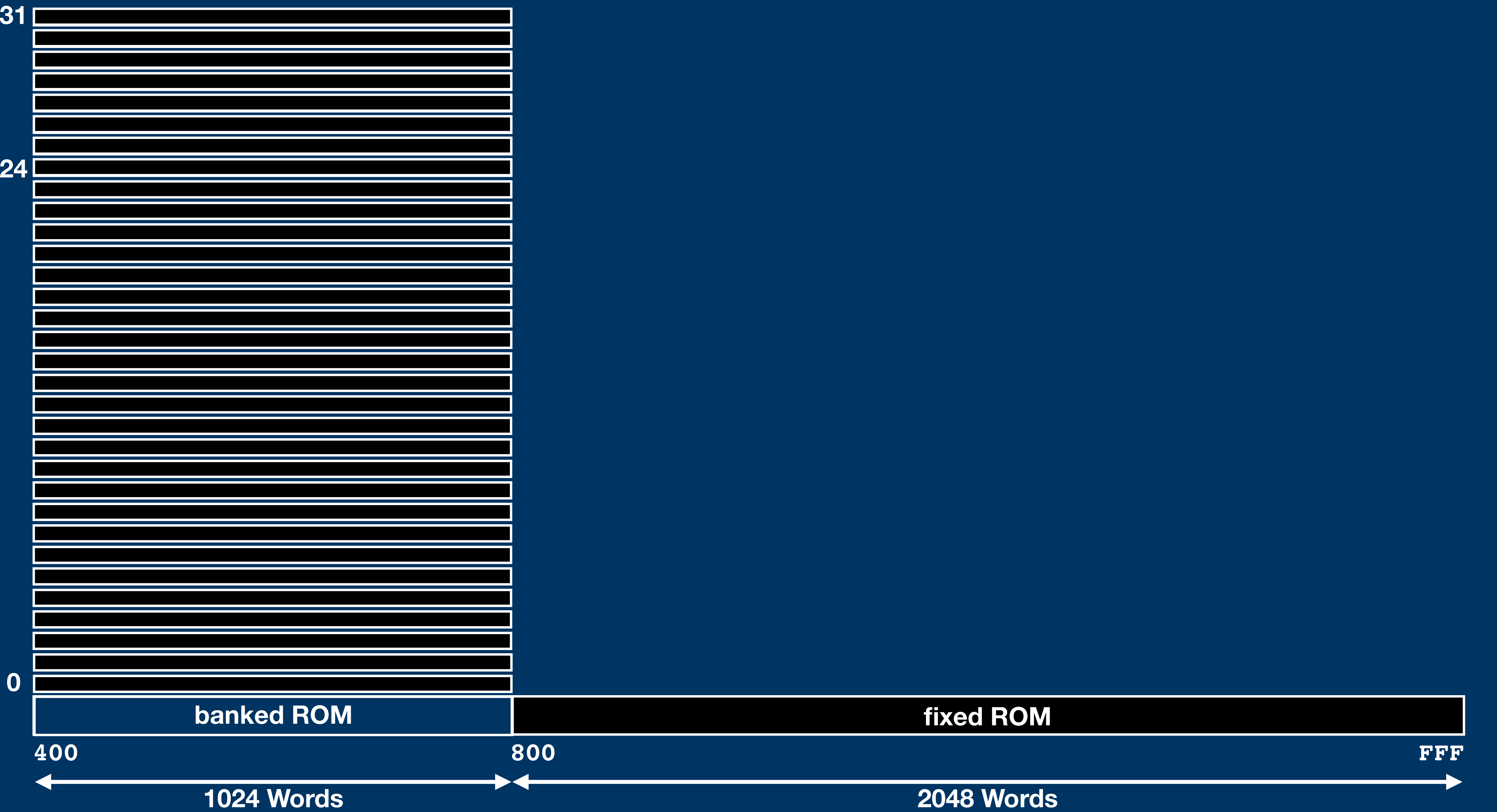
Memory Map: ROM

ROM

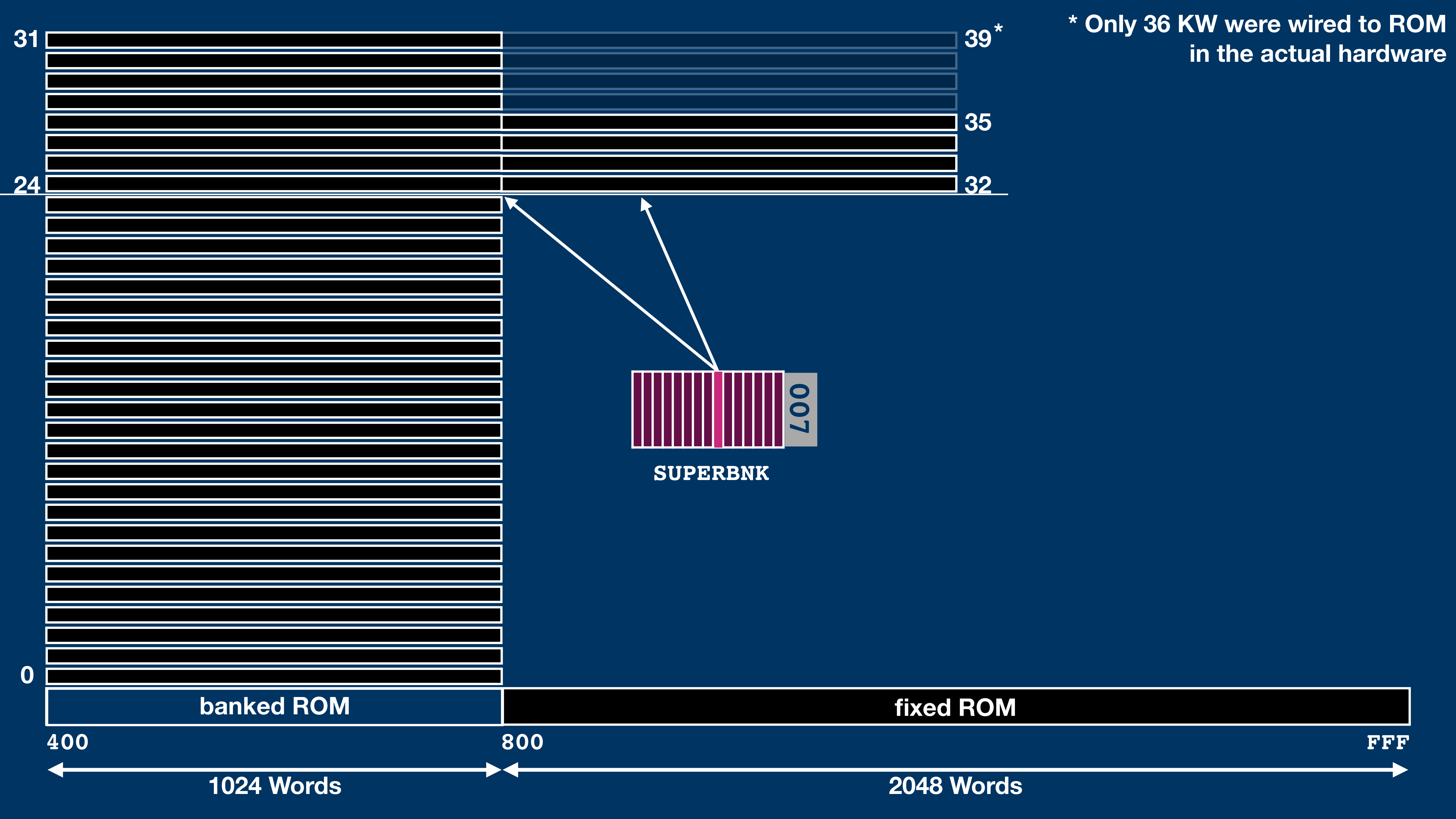
400

FFF

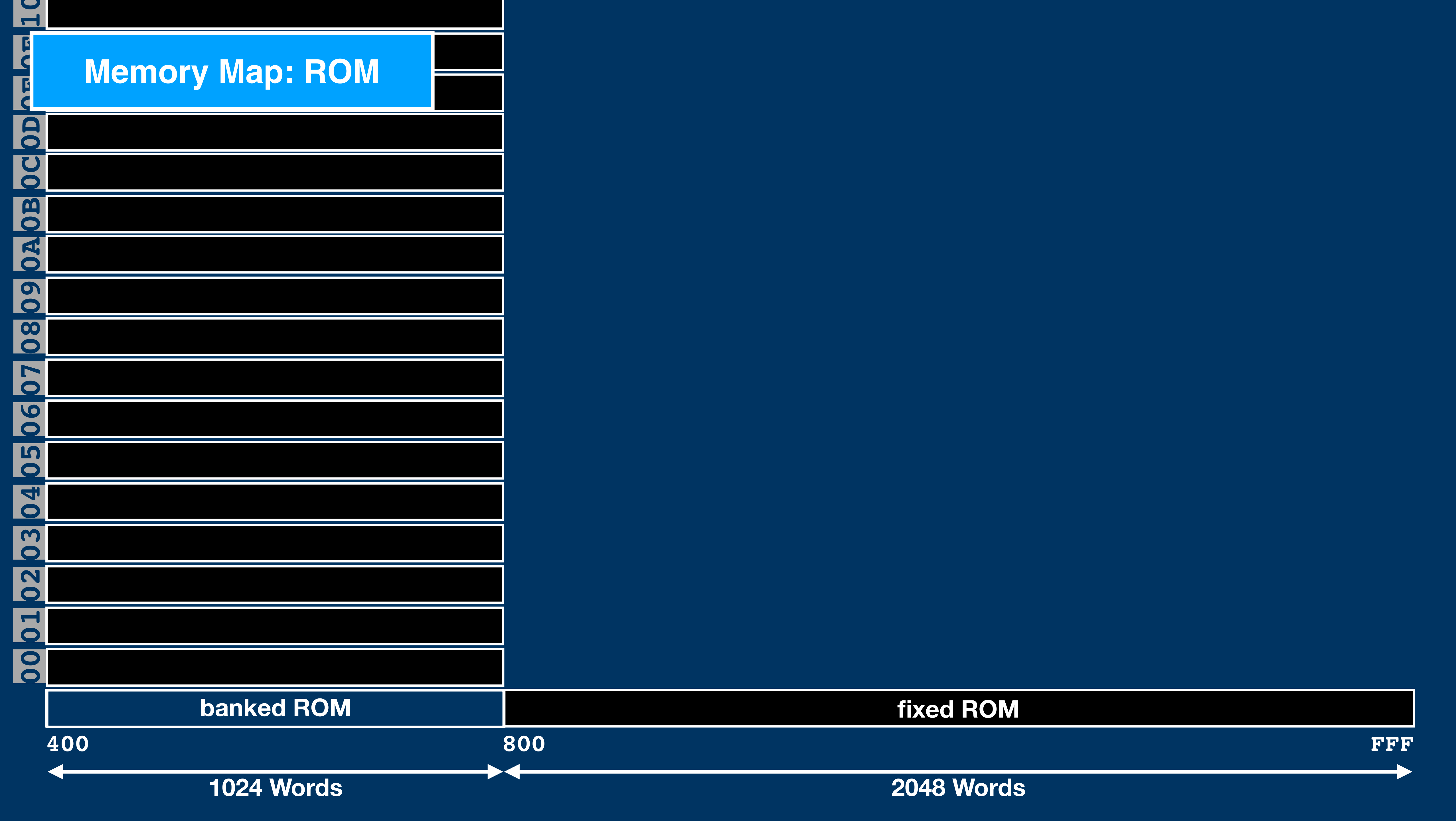


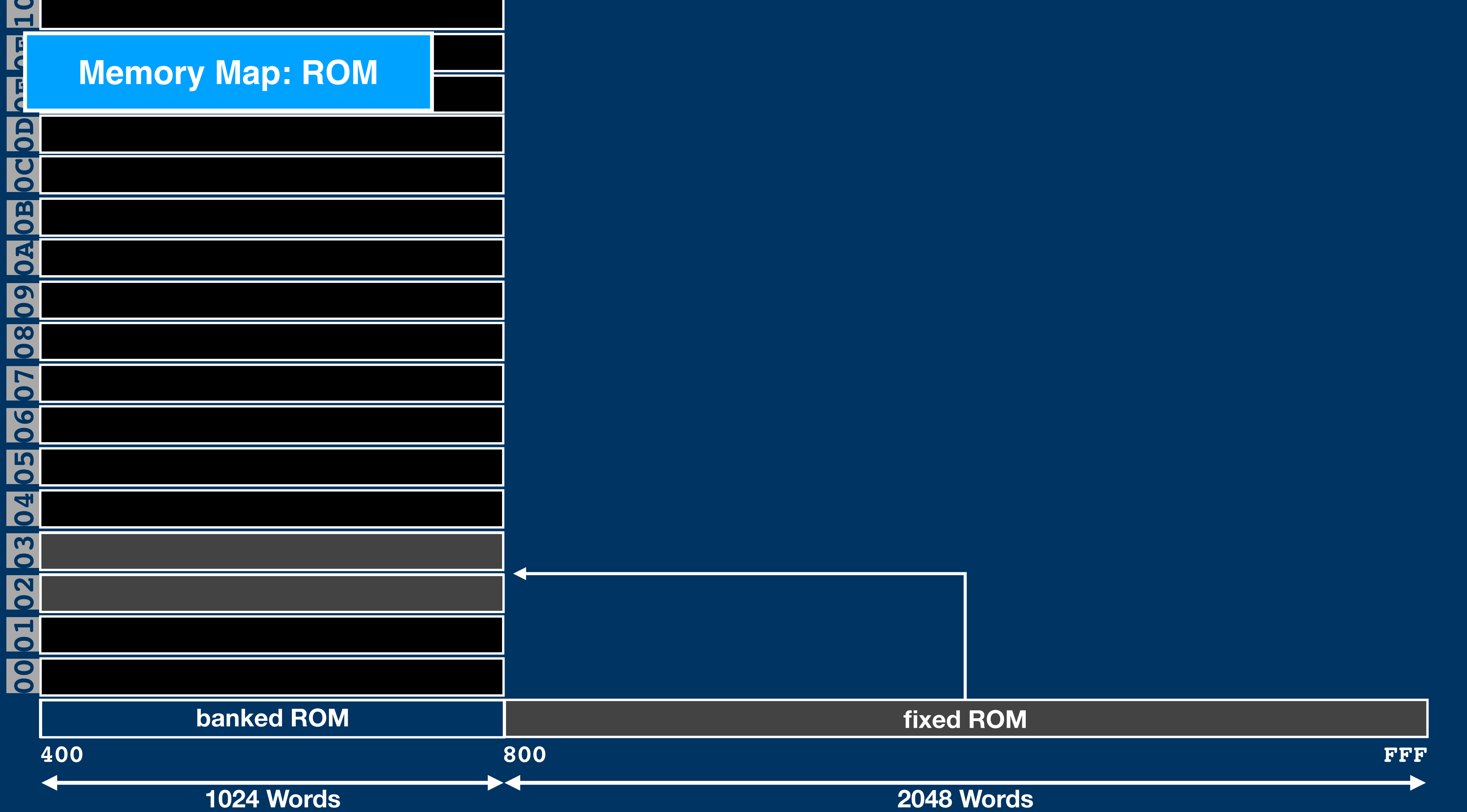






# Memory Map: ROM





Memory Map: ROM

ROM

400

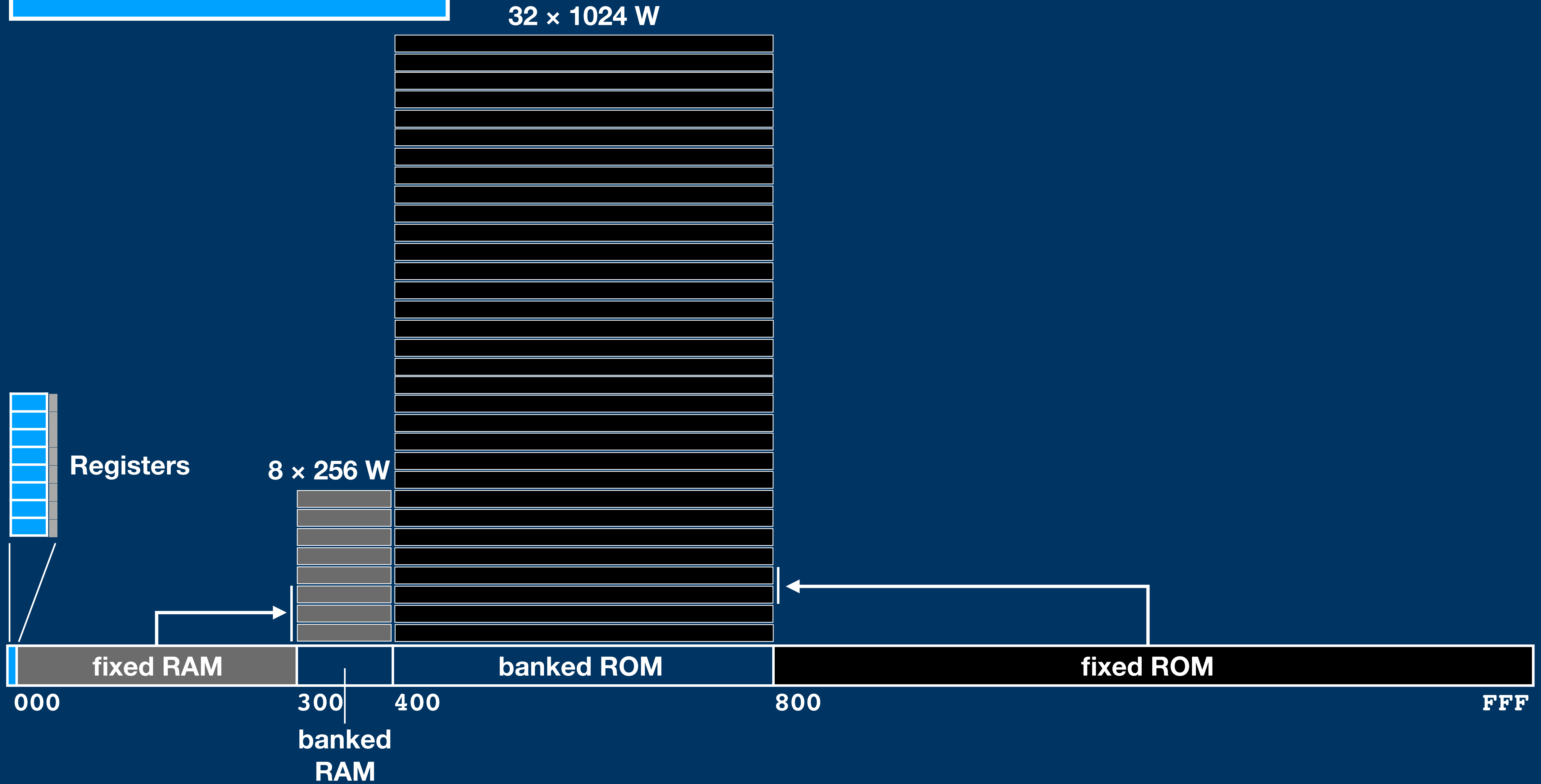
FFF



Memory Map



# Memory Map



## Switch RAM Bank

# Id eb, a



0005



OF00

LR

# 05AA

EB

0000

FB

0000

PC

# 0421

BB

0000

0

0000



---



O

## banked RAM

300

### 3FF

Switch RAM Bank

ld eb, a

Store	ld [k], a		TS k	
	5800+k			2
	M[k] ← A			

A	0005
B	0F00
LR	05AA
EB	0000
FB	0000
PC	0421
BB	0000
0	0000

7	
6	
5	
4	
3	
2	
1	
0	

banked RAM	
300	3FF



Switch RAM Bank

ld [31], a

Store	ld [k], a		TS k	
	5800+k			2
	M[k] ← A			

A	0005	000
B	0F00	001
LR	05AA	002
EB	0000	003
FB	0000	004
PC	0421	005
BB	0000	006
0	0000	007

7	
6	
5	
4	
3	
2	
1	

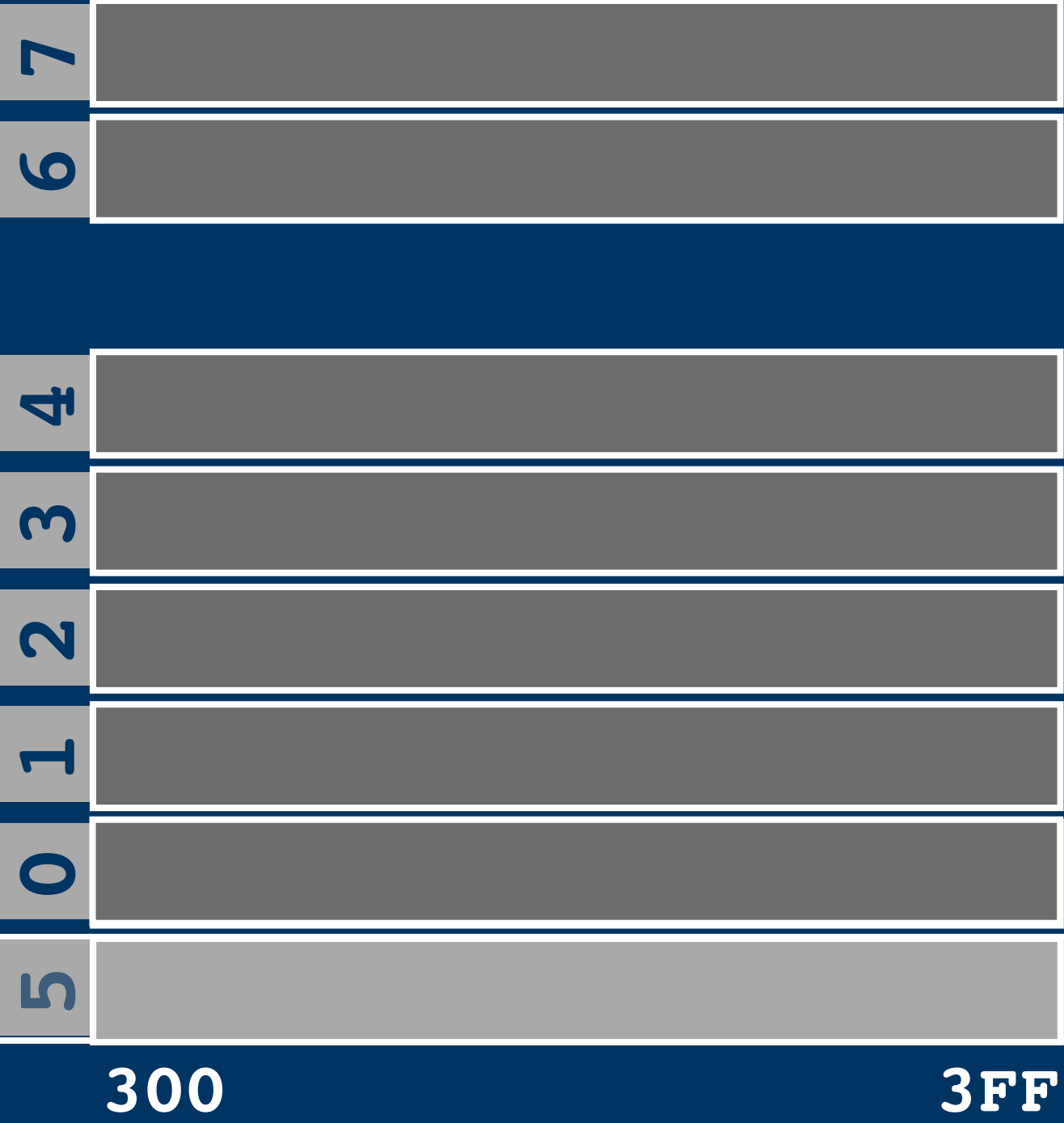


Switch RAM Bank

ld [31], a

Store	ld [k], a		TS k	
	5800+k			2
	M[k] ← A			

A	0005	000
B	0F00	001
LR	05AA	002
EB	0005	003
FB	0000	004
PC	0421	005
BB	0000	006
0	0000	007



Switch ROM Bank

ld fb, a

Store	ld [k], a		TS k	
	5800+k			2
	M[k] ← A			

A	0005	000
B	0F00	001
LR	05AA	002
EB	0000	003
FB	0000	004
PC	041F	005
BB	0000	006
0	0000	007

Switch ROM Bank

ld ~~rb~~<sup>[4]</sup>, a

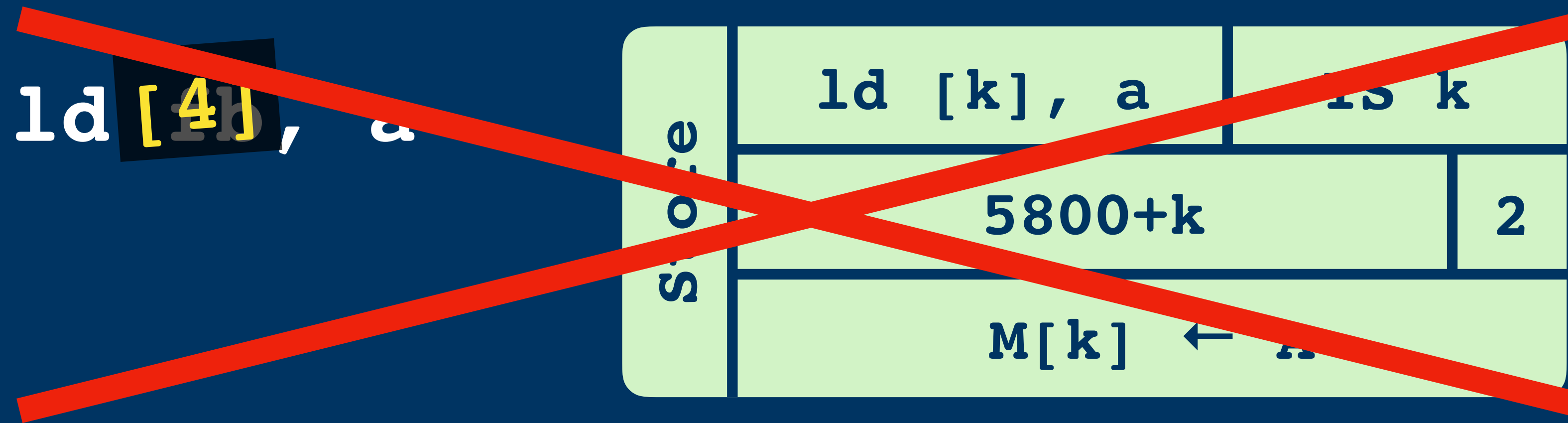
Store	ld [k], a		TS k	
	5800+k			2
	M[k] ← A			

A	0005	000
B	0F00	001
LR	05AA	002
EB	0000	003
FB	0000	004
PC	041F	005
BB	0000	006
0	0000	007



Switch ROM Bank

A	0005	000
B	0F00	001
LR	05AA	002
EB	0000	003
FB	0000	004
PC	041F	005
BB	0000	006
0	0000	007



PC →

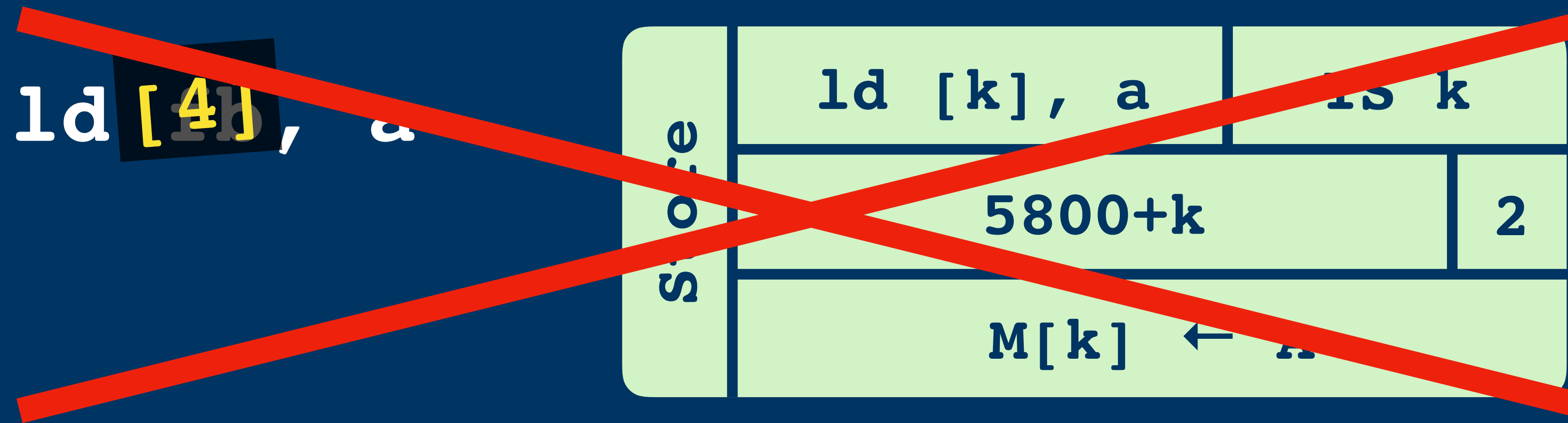
00	
5804	41F
04E5	420
7E00	42

`ld fb, a`

`call $4e5`

Switch ROM Bank

A	0005	000
B	0F00	001
LR	05AA	002
EB	0000	003
FB	0005	004
PC	041F	005
BB	0000	006
0	0000	007



PC →

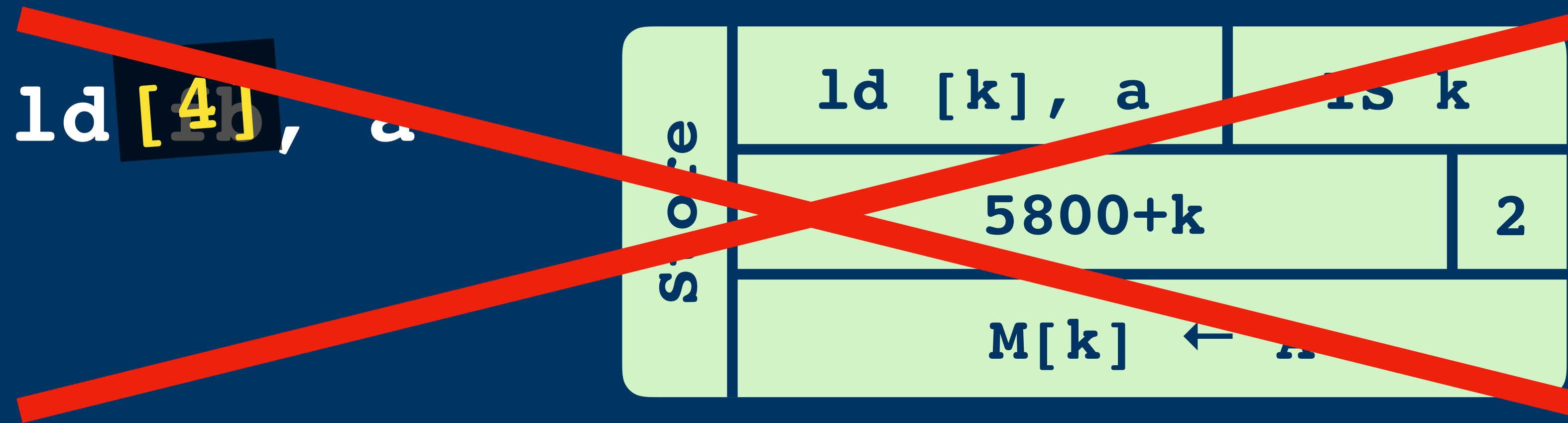
00	
5804	41F
04E5	420
7E00	42

`ld fb, a`

`call $4e5`

Switch ROM Bank

A	0005	000
B	0F00	001
LR	05AA	002
EB	0000	003
FB	0005	004
PC	041F	005
BB	0000	006
0	0000	007



PC →

05	
3FE0	41F
2BFF	420
	42

`ld a, [$fe0]`  
`inc [$3ff]`

Call Far

callf

A	0000	000
B	0000	001
LR	05AA	002
EB	0000	003
FB	0000	004
PC	0421	005
BB	0000	006
0	0000	007

Call Far	callf		DTCF	
	xchg ab, [4]			3
	A ↔ FB B ↔ PC			



Call Far

callf

A	0000	000
B	0000	001
LR	05AA	002
EB	0000	003
FB	0000	004
PC	0421	005
BB	0000	006
0	0000	007

Call Far	callf		DTCF	
	xchg ab, [4]			alias
	A ↔ FB B ↔ PC			3

Call Far

callf

Call Far

callf

DTCF

xchg ab, [4]

alias

3

A ↔ FB

B ↔ PC

A 4000

000

B 0400

001

LR 05AA

002

EB 0000

003

FB 0000

004

PC 0421

005

BB 0000

006

0 0000

007

PC →

3451

41F

ld ab, [\$450]

5205

420

callf

3103

421

ld a, [\$103]

5880

400

ld [\$80], a

0002

401

retf

4

3193

44E

4584

44F

4000

450

0400

451

37B5

452

52A3

453

24D1

454

Call Far

callf

Call Far

callf

DTCF

xchg ab, [4]

alias

3

A ↔ FB

B ↔ PC

A 0000

000

B 0421

001

LR 05AA

002

EB 0000

003

FB 4000

004

PC 0400

005

BB 0000

006

0 0000

007

3451

41F

5205

420

3103

421

5880

400

0002

401

ld ab, [\$450]

callf

ld a, [\$103]

ld [\$80], a

retf

3193

44E

4584

44F

4000

450

0400

451

37B5

452

52A3

453

24D1

454

PC →



Return Far

retf

Return Far

retf

DTCF

xchg ab, [4]

alias

3

A ↔ FB

B ↔ PC

A 0000

000

B 0421

001

LR 05AA

002

EB 0000

003

FB 4000

004

PC 0400

005

BB 0000

006

0 0000

007

3451

41F

5205

420

3103

421

5880

400

0002

401

ld ab, [\$450]

callf

ld a, [\$103]

ld [\$80], a

retf

PC →



Return Far

retf

Return Far

retf

DTCF

xchg ab, [4]

alias

3

A ↔ FB

B ↔ PC

A 4000

000

B 0400

001

LR 05AA

002

EB 0000

003

FB 0000

004

PC 0421

005

BB 0000

006

0 0000

007

PC →

3451

41F

ld ab, [\$450]

5205

420

callf

3103

421

ld a, [\$103]

5880

400

ld [\$80], a

0002

401

retf

# Bank Registers

# Bank Registers

**FB**

**0000**

**EB**

**0000**

Bank Registers

5 bits  
32 banks

FB

f	f	f	f	f	0	0	0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

3 bits  
8 banks

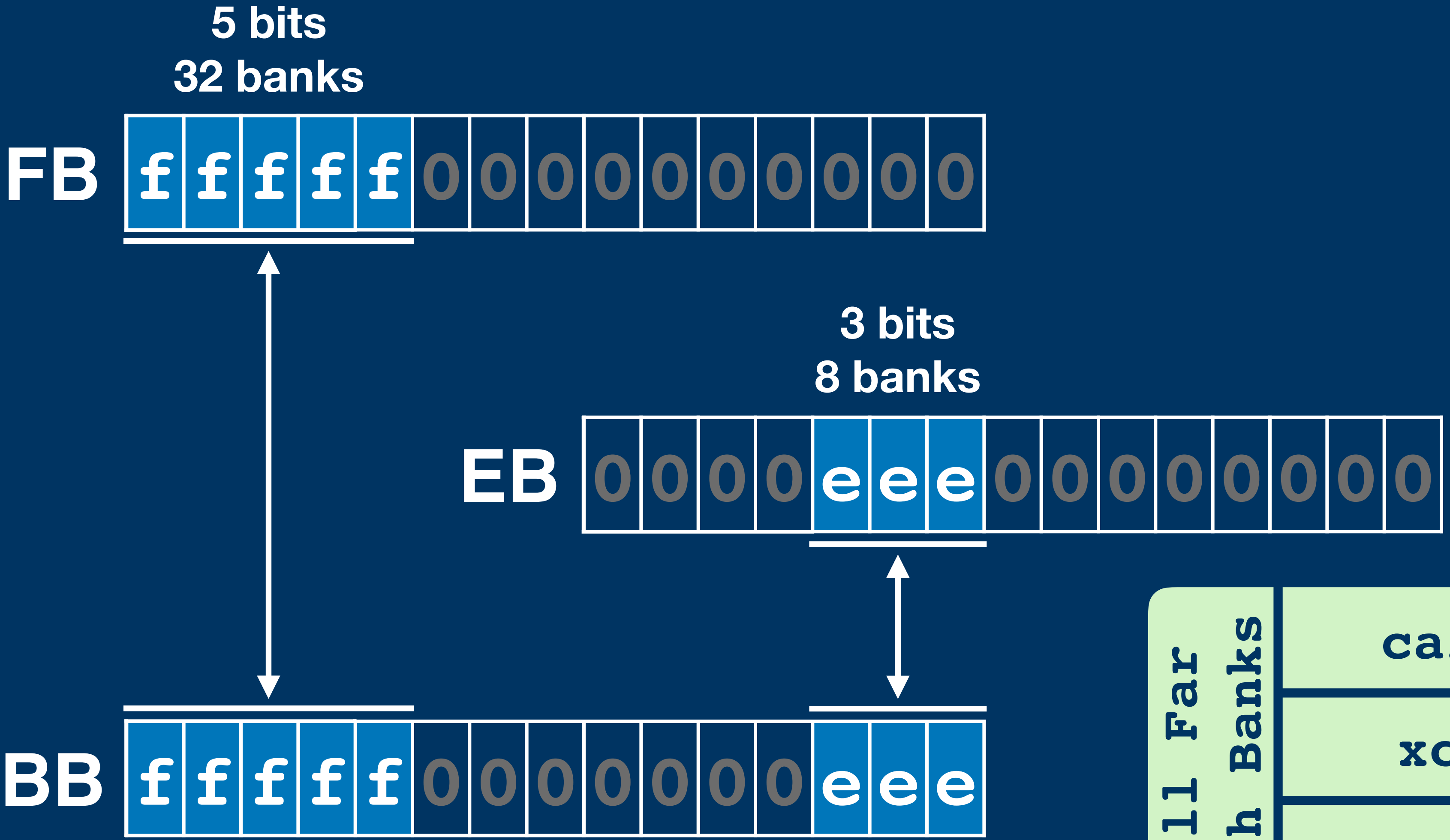
EB

0	0	0	0	e	e	e	0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---





# Bank Registers



Call Far Both Banks	callfbb		DTCB	
	xchg ab, [5]		alias	3
	A ↔ PC B ↔ BB			

# Call Far Both Banks

32 × 1024 W

8 × 256 W

Registers

fixed RAM

banked ROM

fixed ROM

000

300

400

800

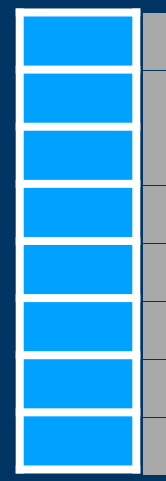
FFF

banked  
RAM

Call Far Both Banks	callfbb		DTCB	
	xchg ab, [5]		alias	3
	A ↔ PC B ↔ BB			

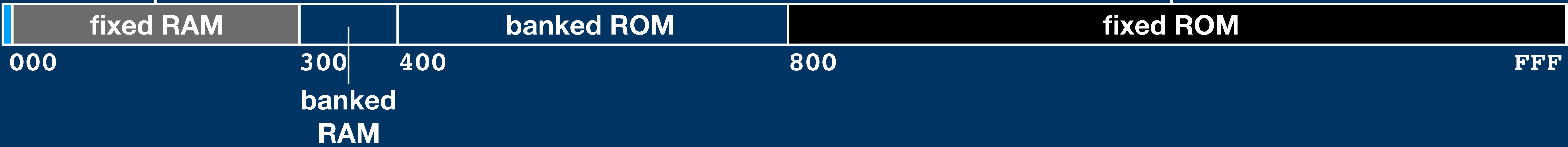
# Call Far Both Banks

32 × 1024 W



Registers

8 × 256 W



000

300

400

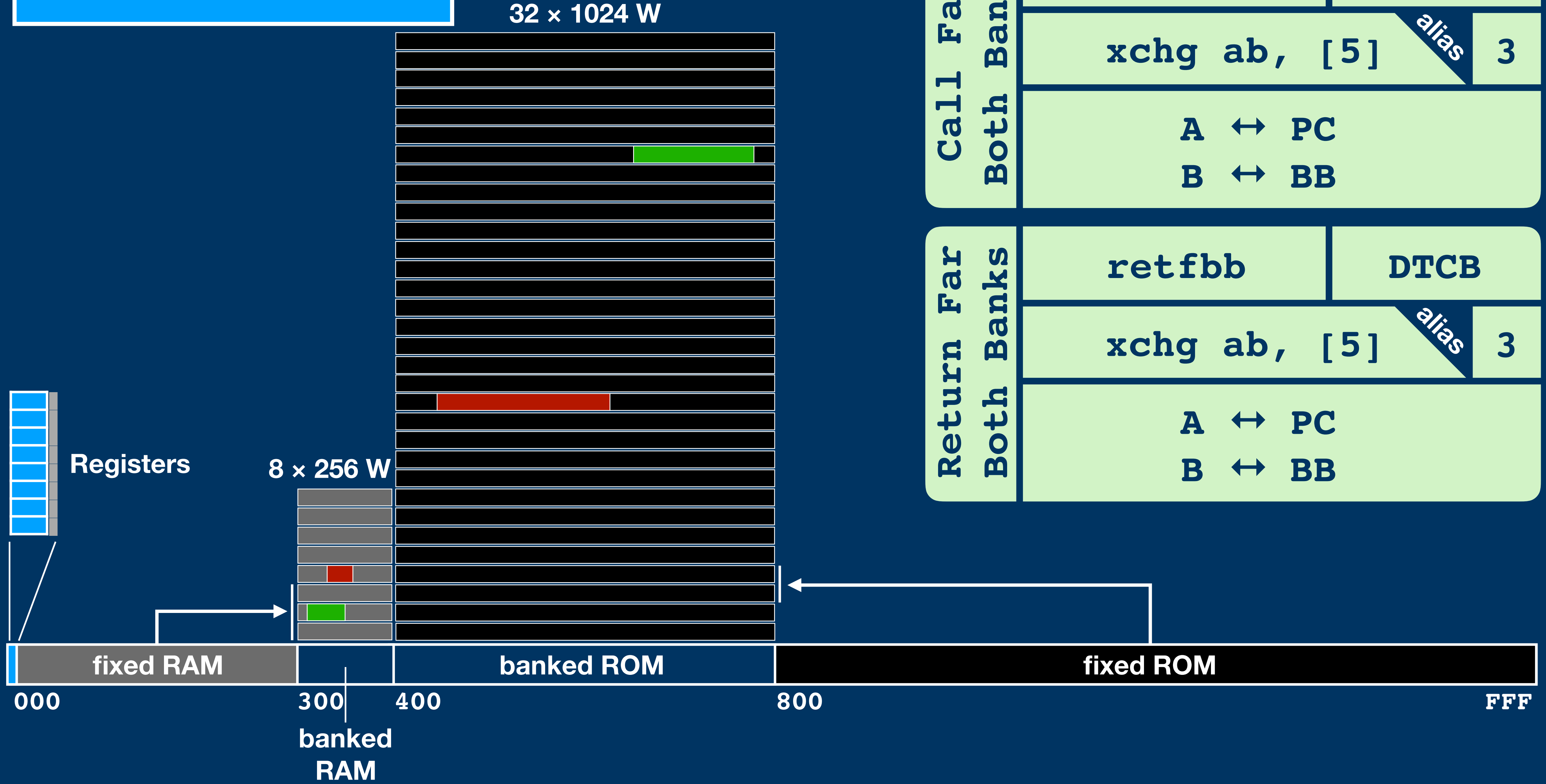
800

FFF

Call Far Both Banks	callfbb		DTCB	
	xchg ab, [5]			alias
	A ↔ PC B ↔ BB			3



# Call Far Both Banks



# Register Ordering

A	0400	000
B	5007	001
LR	05AA	002
EB	0000	003
FB	0000	004
PC	0421	005
BB	0000	006
0	0000	007

# Register Ordering

A	0400	000
B	5007	001
LR	05AA	002
EB	0000	003
FB	0000	004
PC	0421	005
BB	0000	006
0	0000	007

## Register Ordering

A	0400	000	
B	5007	001	
LR	05AA	002	
EB	0000	003	
FB	0000	004	Call Far <code>xchg ab, [4]</code>
PC	0421	005	
BB	0000	006	Call Far Both Banks <code>xchg ab, [5]</code>
0	0000	007	



## Registers

A	0400	000
B	5007	001
LR	05AA	002
EB	0000	003
FB	0000	004
PC	0421	005
BB	0000	006
0	0000	007

## Registers

<b>A</b>	<b>0007</b>	<b>000</b>
<b>B</b>	<b>0F00</b>	<b>001</b>
<b>LR</b>	<b>05AA</b>	<b>002</b>
<b>EB</b>	<b>0000</b>	<b>003</b>
<b>FB</b>	<b>0000</b>	<b>004</b>
<b>PC</b>	<b>0421</b>	<b>005</b>
<b>BB</b>	<b>0000</b>	<b>006</b>
<b>0</b>	<b>0000</b>	<b>007</b>

## Registers

<b>A</b>	<b>0007</b>	<b>000</b>
<b>B</b>	<b>0F00</b>	<b>001</b>
<b>LR</b>	<b>05AA</b>	<b>002</b>
<b>EB</b>	<b>0000</b>	<b>003</b>
<b>FB</b>	<b>0000</b>	<b>004</b>
<b>PC</b>	<b>0421</b>	<b>005</b>
<b>BB</b>	<b>0000</b>	<b>006</b>
<b>0</b>	<b>0000</b>	<b>007</b>

Registers

DD	0000	06
0	0000	07
A'	0000	08
B'	0000	09
LR'	0000	0A
	0000	0B
	0000	0C
PC'	0000	0D
BB'	0000	0E
IR'	0000	0F



Editing

BB	0000	006
0	0000	007
A'	0000	008
B'	0000	009
LR'	0000	00A
	0000	00B
	0000	00C
PC'	0000	00D
BB'	0000	00E
IR'	0000	00F

Editing

DD	0000	E
IR'	0000	00F
ROR	0000	010
SHR	0000	011
ROL	0000	012
SHR7	0000	013

Editing

ROR

0000

010

SHR

0000

011

ROL

0000

012

SHR7

0000

013

Editing

ROR

0000000000000000

SHR

0000

011

ROL

0000

012

SHR7

0000

013



Editing

ROR

010111001001011

SHR

0000

011

ROL

0000

012

SHR7

0000

013

Editing

ROR

1	0	1	0	1	1	1	0	0	1	0	0	1	0	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

SHR

0000

011

ROL

0000

012

SHR7

0000

013

Editing

ROR

1	0	1	0	1	1	1	0	0	1	0	0	1	0	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

SHR

0	0	0	0
---	---	---	---

011

ROL

1	0	1	1	1	0	0	1	0	0	1	0	1	1	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

SHR7

0	0	0	0
---	---	---	---

013

Editing

ROR

1	0	1	0	1	1	1	0	0	1	0	0	1	0	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

SHR

0	0	1	0	1	1	1	0	0	1	0	0	1	0	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

ROL

1	0	1	1	1	0	0	1	0	0	1	0	1	1	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

SHR7

0	0	0	0
---	---	---	---

013



Editing

ROR

1	0	1	0	1	1	1	0	0	1	0	0	1	0	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

SHR

0	0	1	0	1	1	1	0	0	1	0	0	1	0	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

ROL

1	0	1	1	1	0	0	1	0	0	1	0	1	1	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

SHR7

0	0	0	0	0	0	0	0	1	0	1	1	1	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

## I/O Channels

I/O Channels

RAM

ROM

CPU



I/O Channels

Memory

CPU





I/O Channels

Memory

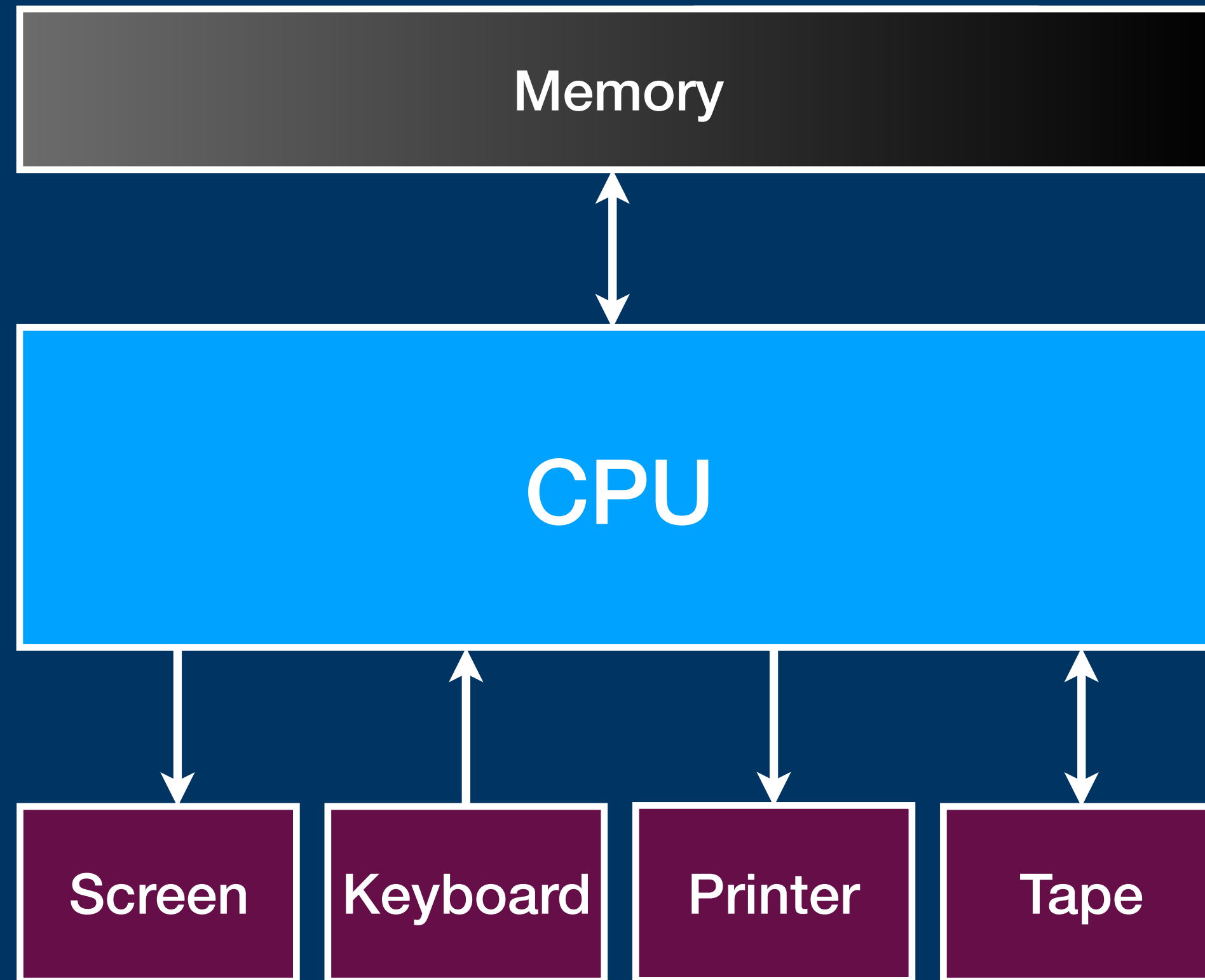
CPU

Screen

Keyboard

Printer

Tape



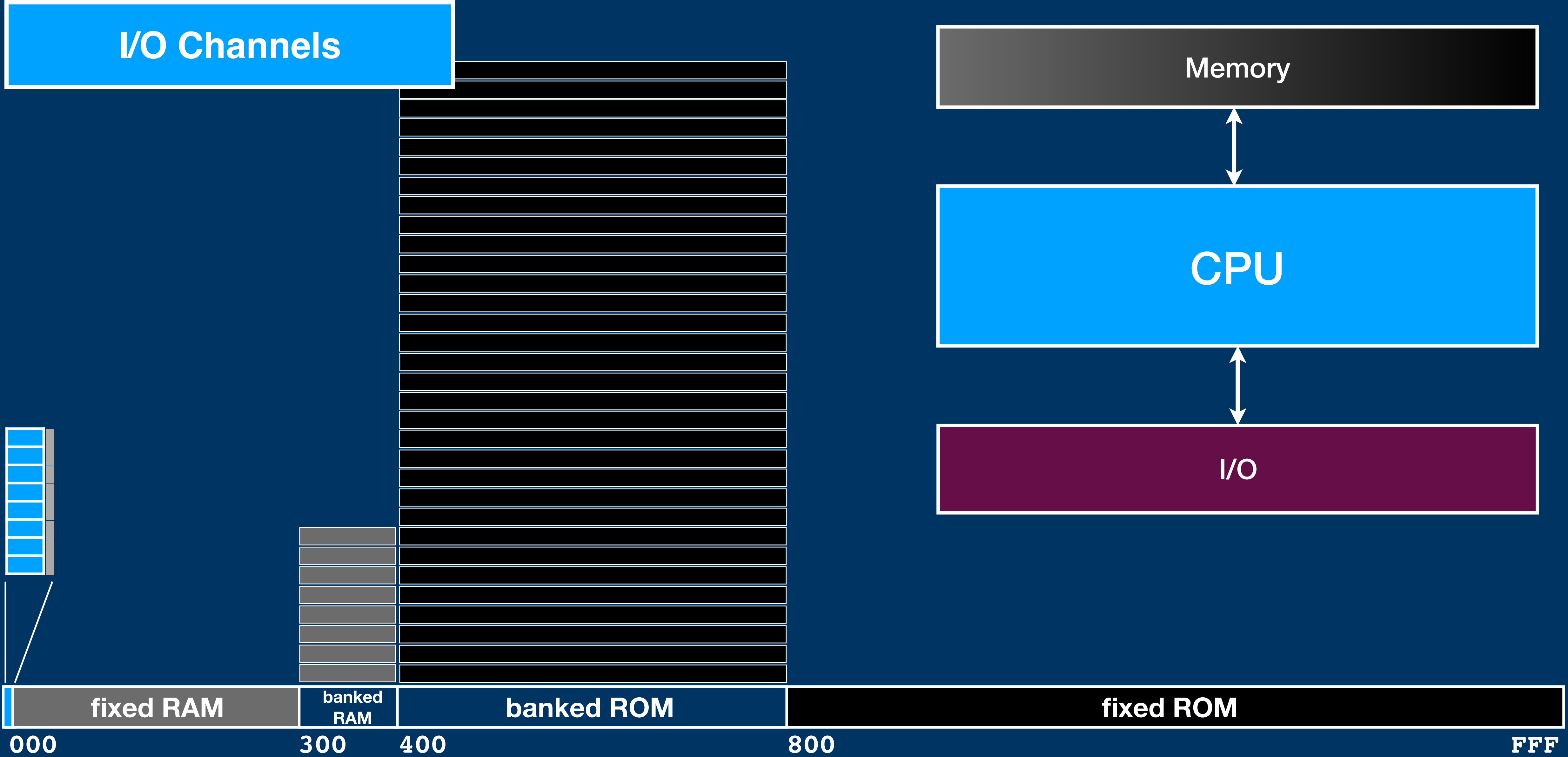
I/O Channels

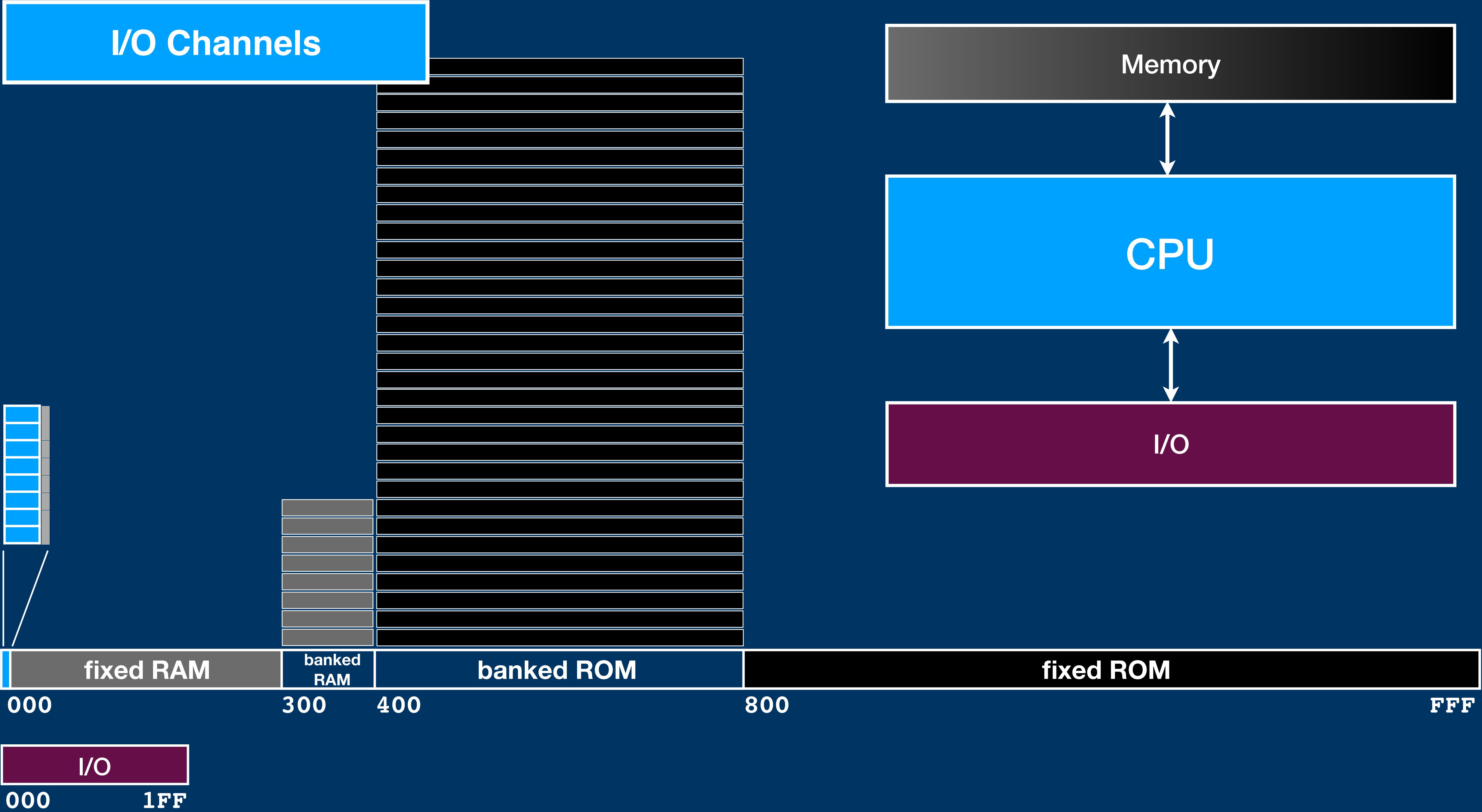
Memory

CPU

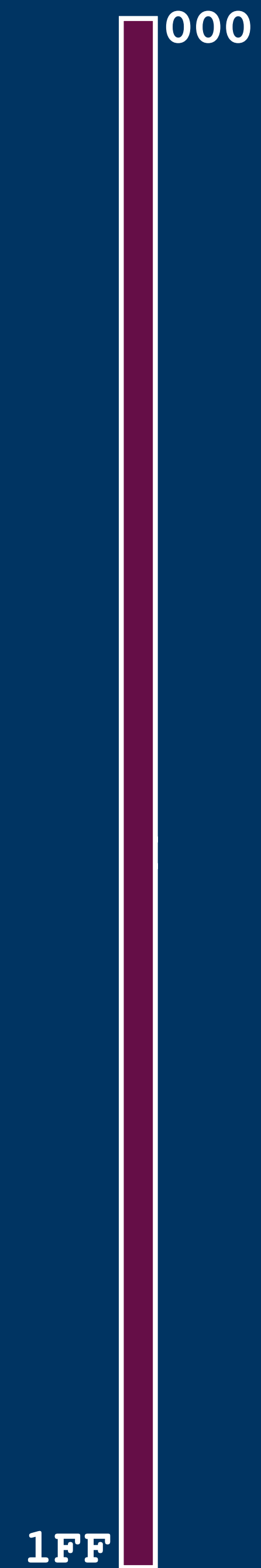
I/O

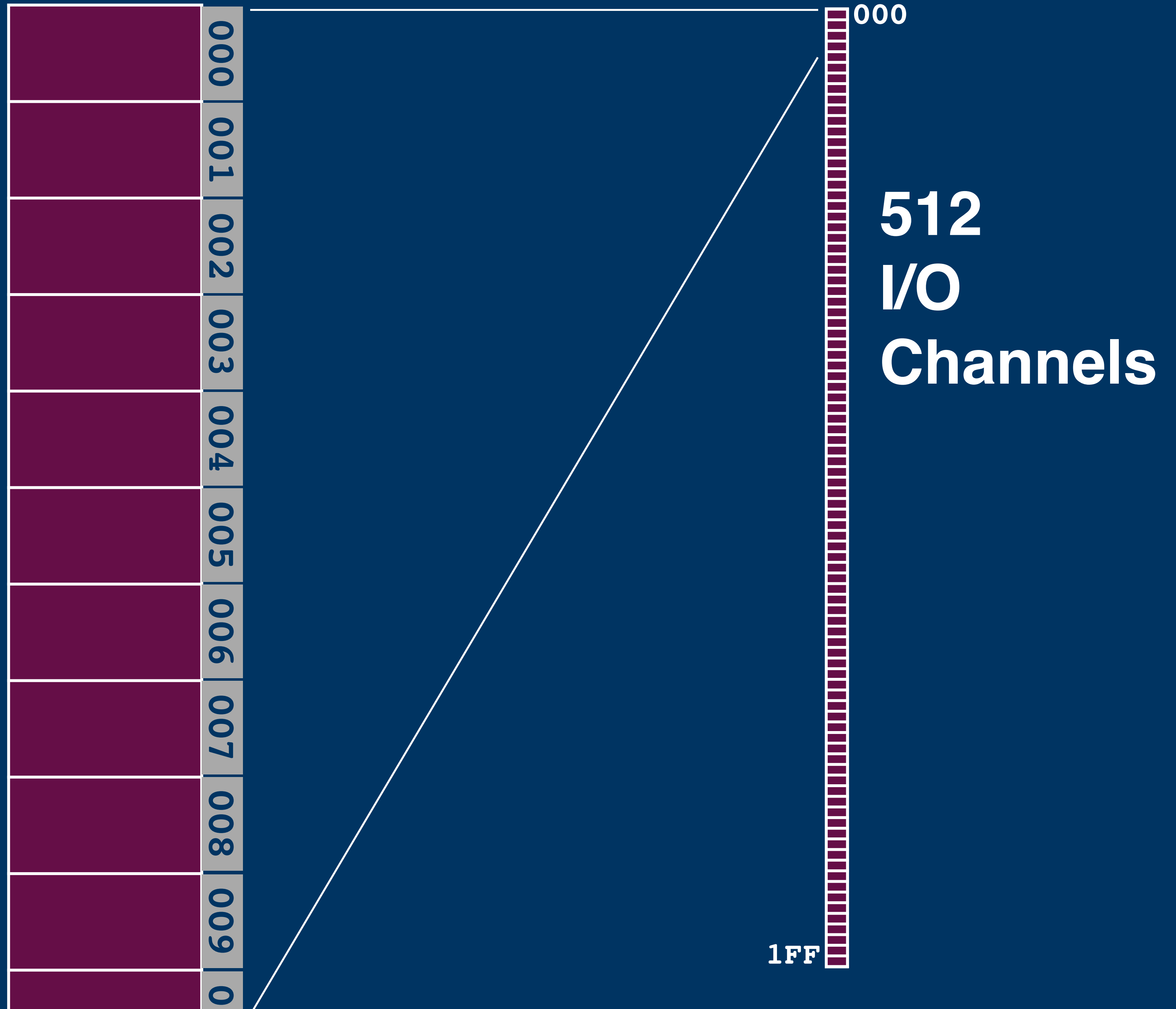












I/O Channels

000	
001	
002	
003	
004	
005	
006	
007	
008	
009	
00	

I/O Channels

In	in a, [kc]	READ kc
	0006, 0000+kc	3
	A ← IO[kc]	
Out	out [kc], a	WRITE kc
	0006, 0200+kc	3
	IO[kc] ← A	

	000
	001
	002
	003
	004
	005
	006
	007
	008
	009
	00

15 bit



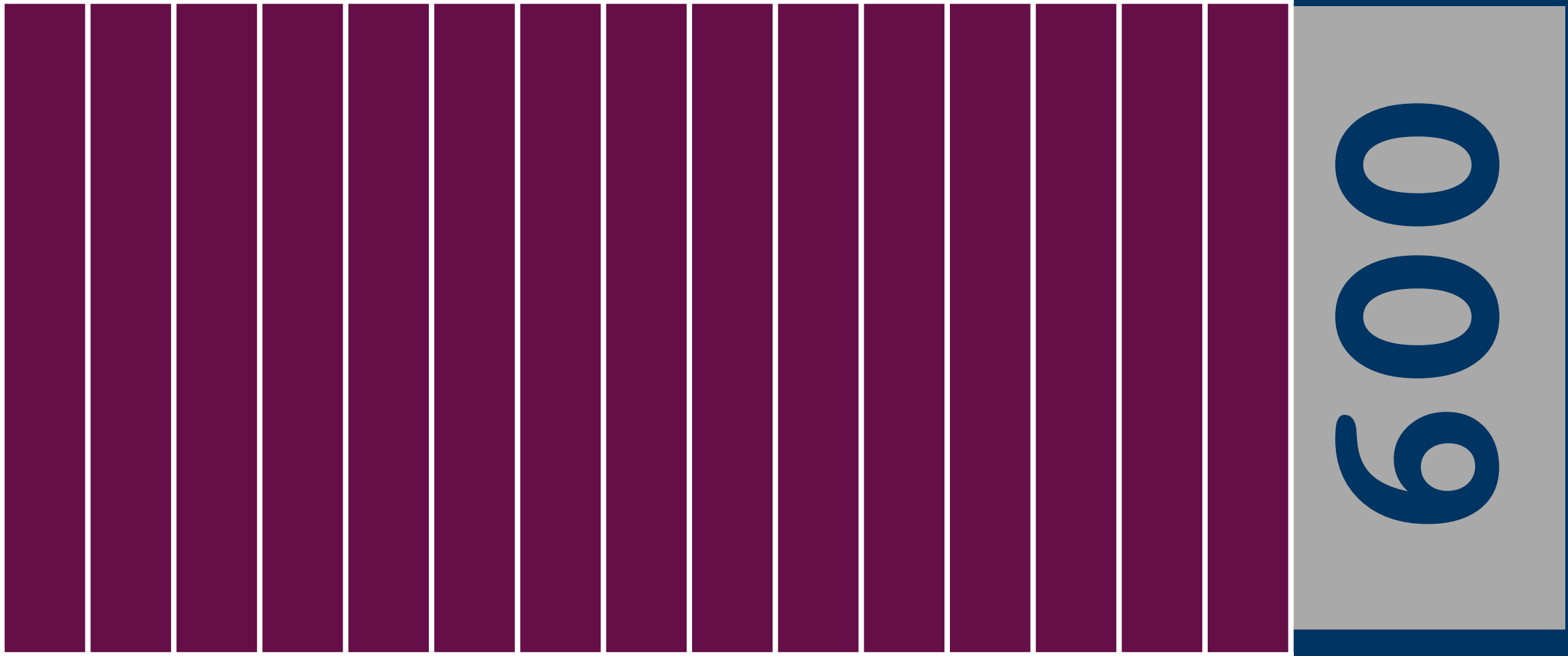
I/O Channels

In	in a, [kc]	READ kc
	0006, 0000+kc	3
	A ← IO[kc]	
Out	out [kc], a	WRITE kc
	0006, 0200+kc	3
	IO[kc] ← A	

	000
	001
	002
	003
	004
	005
	006
	007
	008
	009
	00

15 bit

I/O Channels



009

ISS Warning
Lamp: COMP ACTY
Lamp: UPLINK ACTY
Lamp: TEMP
Lamp: KEY REL
Lamp: VERB/NOUN
Lamp: OPR ERR
Test Connector
Caution Reset
Engine On/Off

I/O Channels

I/O Channels

009

ISS Warning

Lamp: COMP ACTY

Lamp: UPLINK ACTY

Lamp: TEMP

Lamp: KEY REL

Lamp: VERB/NOUN

Lamp: OPR ERR

Test Connector

Caution Reset

Engine On/Off

UPLINK ACTY	TEMP
NO ATT	GIMBAL LOCK
STBY	PROG
KEY REL	RESTART
OPR ERR	TRACKER
	ALT
	VEL

COMP ACTY

PROG

77

VERB

NOUN

31

33

÷58944

÷35870

-836 14

VERB

NOUN

+

-

0

1

2

3

4

5

6

7

8

9

CLR

PRO

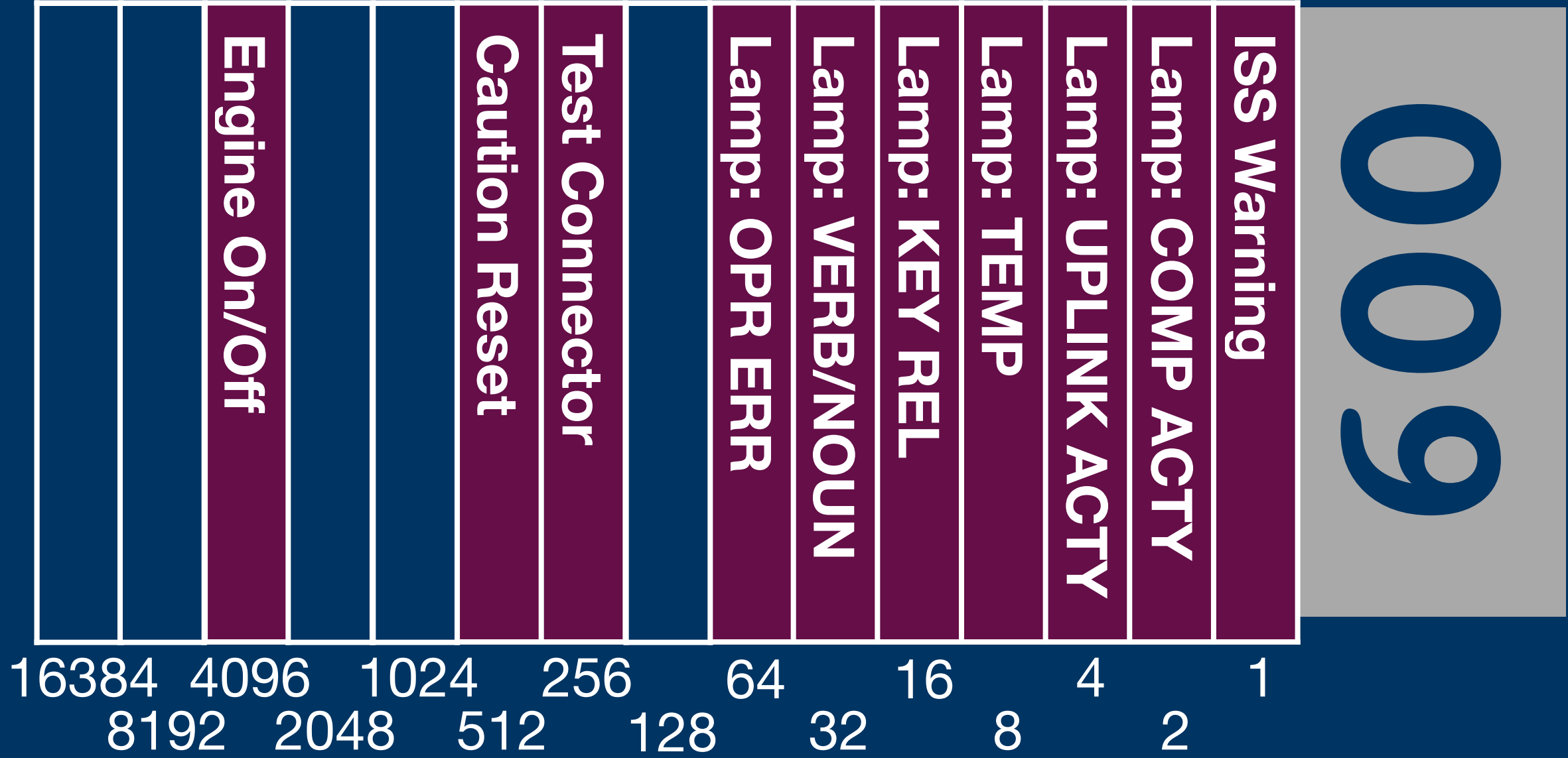
KEY REL

ENTR

RSET



I/O Channels



ld a, #2

out| [9]

ld a, #~2

PC →

out& [9]

**Out AND**

out& [kc], a	WAND kc
0006, 0600+kc	3
A, IO[kc] ← A & IO[kc]	

**Out OR**

out  [kc], a	WOR kc
0006, 0A00+kc	3
A, IO[kc] ← A   IO[kc]	

# I/O Channels

Out AND	out& [kc], a	WAND kc
	0006, 0600+kc	3
	A, IO[kc] ← A & IO[kc]	
Out OR	out  [kc], a	WOR kc
	0006, 0A00+kc	3
	A, IO[kc] ← A   IO[kc]	

# I/O Channels

In	in a, [kc]		READ kc
	0006, 0000+kc		3
	A ← IO[kc]		

Out	out [kc], a		WRITE kc
	0006, 0200+kc		3
	IO[kc] ← A		

Out AND	out& [kc], a		WAND kc
	0006, 0600+kc		3
	A, IO[kc] ← A & IO[kc]		

Out OR	out  [kc], a		WOR kc
	0006, 0A00+kc		3
	A, IO[kc] ← A   IO[kc]		

# I/O Channels

In	in a, [kc]	READ kc
	0006, 0000+kc	3
	A ← IO[kc]	

In AND	in& a, [kc]	RAND kc
	0006, 0400+kc	3
	A ← A & IO[kc]	

Out	out [kc], a	WRITE kc
	0006, 0200+kc	3
	IO[kc] ← A	

Out AND	out& [kc], a	WAND kc
	0006, 0600+kc	3
	A, IO[kc] ← A & IO[kc]	

Out OR	out  [kc], a	WOR kc
	0006, 0A00+kc	3
	A, IO[kc] ← A   IO[kc]	



# I/O Channels

In	in a, [kc]	READ kc
	0006, 0000+kc	3
	A ← IO[kc]	

In AND	in& a, [kc]	RAND kc
	0006, 0400+kc	3
	A ← A & IO[kc]	

In OR	in  a, [kc]	ROR kc
	0006, 0800+kc	3
	A ← A   IO[kc]	

Out	out [kc], a	WRITE kc
	0006, 0200+kc	3
	IO[kc] ← A	

Out AND	out& [kc], a	WAND kc
	0006, 0600+kc	3
	A, IO[kc] ← A & IO[kc]	

Out OR	out  [kc], a	WOR kc
	0006, 0A00+kc	3
	A, IO[kc] ← A   IO[kc]	

# I/O Channels

In	in a, [kc]	READ kc
	0006, 0000+kc	3
	A ← IO[kc]	

In AND	in& a, [kc]	RAND kc
	0006, 0400+kc	3
	A ← A & IO[kc]	

In OR	in  a, [kc]	ROR kc
	0006, 0800+kc	3
	A ← A   IO[kc]	

Out	out [kc], a	WRITE kc
	0006, 0200+kc	3
	IO[kc] ← A	

Out AND	out& [kc], a	WAND kc
	0006, 0600+kc	3
	A, IO[kc] ← A & IO[kc]	

Out OR	out  [kc], a	WOR kc
	0006, 0A00+kc	3
	A, IO[kc] ← A   IO[kc]	

# I/O Channels

In	in a, [kc]	READ kc
	0006, 0000+kc	3
	A ← IO[kc]	

Out	out [kc], a	WRITE kc
	0006, 0200+kc	3
	IO[kc] ← A	

In AND	in& a, [kc]	RAND kc
	0006, 0400+kc	3
	A ← A & IO[kc]	

Out AND	out& [kc], a	WAND kc
	0006, 0600+kc	3
	A, IO[kc] ← A & IO[kc]	

In OR	in  a, [kc]	ROR kc
	0006, 0800+kc	3
	A ← A   IO[kc]	

Out OR	out  [kc], a	WOR kc
	0006, 0A00+kc	3
	A, IO[kc] ← A   IO[kc]	

In XOR	in^ a, [kc]	RXOR kc
	0006, 0C00+kc	3
	A ← A ^ IO[kc]	

I/O Channels

B  
LR

	000
	001
	002
	003
	004
	005
	006
	007
	008

In	in a, [kc]	READ kc
	0006, 0000+kc	3
	A ← IO[kc]	

In AND	in& a, [kc]	RAND kc
	0006, 0400+kc	3
	A ← A & IO[kc]	

In OR	in  a, [kc]	ROR kc
	0006, 0800+kc	3
	A ← A   IO[kc]	

In XOR	in^ a, [kc]	RXOR kc
	0006, 0C00+kc	3
	A ← A ^ IO[kc]	

Out	out [kc], a	WRITE kc
	0006, 0200+kc	3
	IO[kc] ← A	

Out AND	out& [kc], a	WAND kc
	0006, 0600+kc	3
	A, IO[kc] ← A & IO[kc]	

Out OR	out  [kc], a	WOR kc
	0006, 0A00+kc	3
	A, IO[kc] ← A   IO[kc]	



# Boolean Logic

## Boolean Logic

`and a, b`  $\equiv$  `in & a, [1]`

`or a, b`  $\equiv$  `in | a, [1]`

`xor a, b`  $\equiv$  `in ^ a, [1]`

# Boolean Logic

`and a, b`  $\equiv$  `in& a, [1]`

`or a, b`  $\equiv$  `in| a, [1]`

`xor a, b`  $\equiv$  `in^ a, [1]`

AND	and a, [k]		MASK k	
	7000+k			2
	A ← A & M[k]			

## Counters

A	0007	000
B	0F00	001
LR	05AA	002
EB	0000	003
FB	0000	004
PC	0421	005
BB	0000	006
0	0000	007

## Counters



## Counters

Counters

0

Registers

8

Shadow

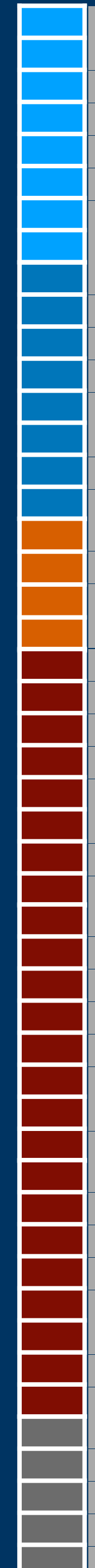
\$10

Editing

\$14

Counters

\$31



# Counters

TIME2	0000	014
TIME1	0000	015
TIME3	0000	016
TIME4	0000	017
TIME5	0000	018
TIME6	0000	019

0  
8  
\$10  
\$14  
\$31

Registers

Shadow

Editing

Counters

0003	028
0000	029
0000	02A

CDUXCMD

CDUYCMD

CDUZCMD

# Counters

TIME2	0000	014
TIME1	0003	015
TIME3	0000	016
TIME4	0000	017
TIME5	0000	018
TIME6	0000	019

0  
8  
\$10  
\$14  
\$31

Registers

Shadow

Editing

Counters

0003	028
0000	029
0000	02A

CDUXCMD

CDUYCMD

CDUZCMD

# Counters

TIME2	0000	014
TIME1	0003	015
TIME3	0000	016
TIME4	0000	017
TIME5	0000	018
TIME6	0000	019

0  
8  
\$10  
\$14  
\$31

Registers

Shadow

Editing

Counters

0000	028
0000	029
0000	02A

CDUXCMD

CDUYCMD

CDUZCMD



Interrupts

A	0000
B	0F00
LR	05AA
EB	0000
FB	0000
PC	041F
BB	0000
0	0000

Interrupts

A	0000
B	0F00
LR	05AA
EB	0000
FB	0000
PC	041F
BB	0000
0	0000

PC →

0000	E
3FE0	41F
5920	420
2102	42

ld a, [\$FE0]

ld [\$0120], a

Interrupts

A	0000
B	0F00
LR	05AA
EB	0000
FB	0000
PC	041F
BB	0000
0	0000

PC →

0000	E
3FE0	41F
5920	420
2102	42

ld a, [\$FE0]

ld [\$0120], a

IR

3FE0

Interrupts

A	0000
B	0F00
LR	05AA
EB	0000
FB	0000
PC	0420
BB	0000
0	0000

PC →

0000	E
3FE0	41F
5920	420
2102	42

ld a, [\$FE0]

ld [\$0120], a

IR 5920

Interrupts

Interrupt



A	0000
B	0F00
LR	05AA
EB	0000
FB	0000
PC	0420
BB	0000
0	0000

PC →

0000	E
3FE0	41F
5920	420
2102	42

ld a, [\$FE0]

ld [\$0120], a

A'	0000
B'	0000
LR'	0000
	0000
	0000
PC'	0000
BB'	0000
IR'	0000

IR 5920



Interrupts

Interrupt



A	0000
B	0F00
LR	05AA
EB	0000
FB	0000
PC	0420
BB	0000
0	0000

PC →

0000	E
3FE0	41F
5920	420
2102	42

ld a, [\$FE0]

ld [\$0120], a

A'	0000
B'	0000
LR'	0000
	0000
	0000
PC'	0420
BB'	0000
IR'	5920

Interrupts

A	0000
B	0F00
LR	05AA
EB	0000
FB	0000
PC	0814
BB	0000
0	0000

PC →

0000	E
3FE0	41F
5920	420
0000	42
0000	3
500F	814
0000	81

ld a, [

ld [\$0120], a

iret

Int Return	iret		RESUME	
	0017		2	
	(enable int) PC ← PC' IR ← IR'			
20], a			LR'	0000
				0000
				0000
			PC'	0420
			BB'	0000
			IR'	5920

## Interrupts

**B** | **OF00**

EB 0000

PC 0420

0 | 0000

5920



3FeO

5920

2103 42



500F

3103 | 81

# 1d a, [

1d [\$0120], a

# iret

## Int Return

```
iret
```

# RESUME

0017

2

```
(enable int)
  PC ← PC'
  IR ← IR'
```

$$\mathbf{IR} \leftarrow \mathbf{IR}'$$

LR'

0000

0000

0000

PC'

# 0420

BB'

0000

IR'

5920

Interrupts

A	0007	000
B	0F00	001
LR	05AA	002
EB	0000	003
FB	0000	004
PC	0421	005
BB	0000	006
0	0000	007

Interrupts

A	0007	000
B	0F00	001
LR	05AA	002
EB	0000	003
FB	0000	004
PC	0421	005
BB	0000	006
0	0000	007

A'	0000	008
B'	0000	009
LR'	0000	00A
	0000	00B
	0000	00C
PC'	0000	00D
BB'	0000	00E
IR'	0000	00F



Interrupts

A	0007	000
B	0F00	001
LR	05AA	002
EB	0000	003
FB	0000	004
PC	0421	005
BB	0000	006
0	0000	007

A'	0000	008
B'	0000	009
LR'	0000	00A
	0000	00B
	0000	00C
PC'	0000	00D
BB'	0000	00E
IR'	0000	00F

# Interrupts

Saved  
&  
Restored  
in *Hardware*

A	0007	000	A'	0000	008
B	0F00	001	B'	0000	009
LR	05AA	002	LR'	0000	00A
EB	0000	003		0000	00B
FB	0000	004		0000	00C
PC	0421	005	PC'	0000	00D
BB	0000	006	BB'	0000	00E
0	0000	007	IR'	0000	00F

# Interrupts

Saved  
&  
Restored  
in *Hardware*

Reserved  
for Saving &  
Restoring  
in *Software*

Unused

A	0007	000	A'	0000	008
B	0F00	001	B'	0000	009
LR	05AA	002	LR'	0000	00A
EB	0000	003		0000	00B
FB	0000	004		0000	00C
PC	0421	005	PC'	0000	00D
BB	0000	006	BB'	0000	00E
0	0000	007	IR'	0000	00F

# Interrupts

Saved  
&  
Restored  
in *Hardware*

Reserved  
for Saving &  
Restoring  
in *Software*

Unused

V	A	0007	000	A'	0000	008
	B	0F00	001	B'	0000	009
	LR	05AA	002	LR'	0000	00A
	EB	0000	003		0000	00B
	FB	0000	004		0000	00C
	PC	0421	005	PC'	0000	00D
	BB	0000	006	BB'	0000	00E
0		0000	007	IR'	0000	00F

# Interrupts & Overflow

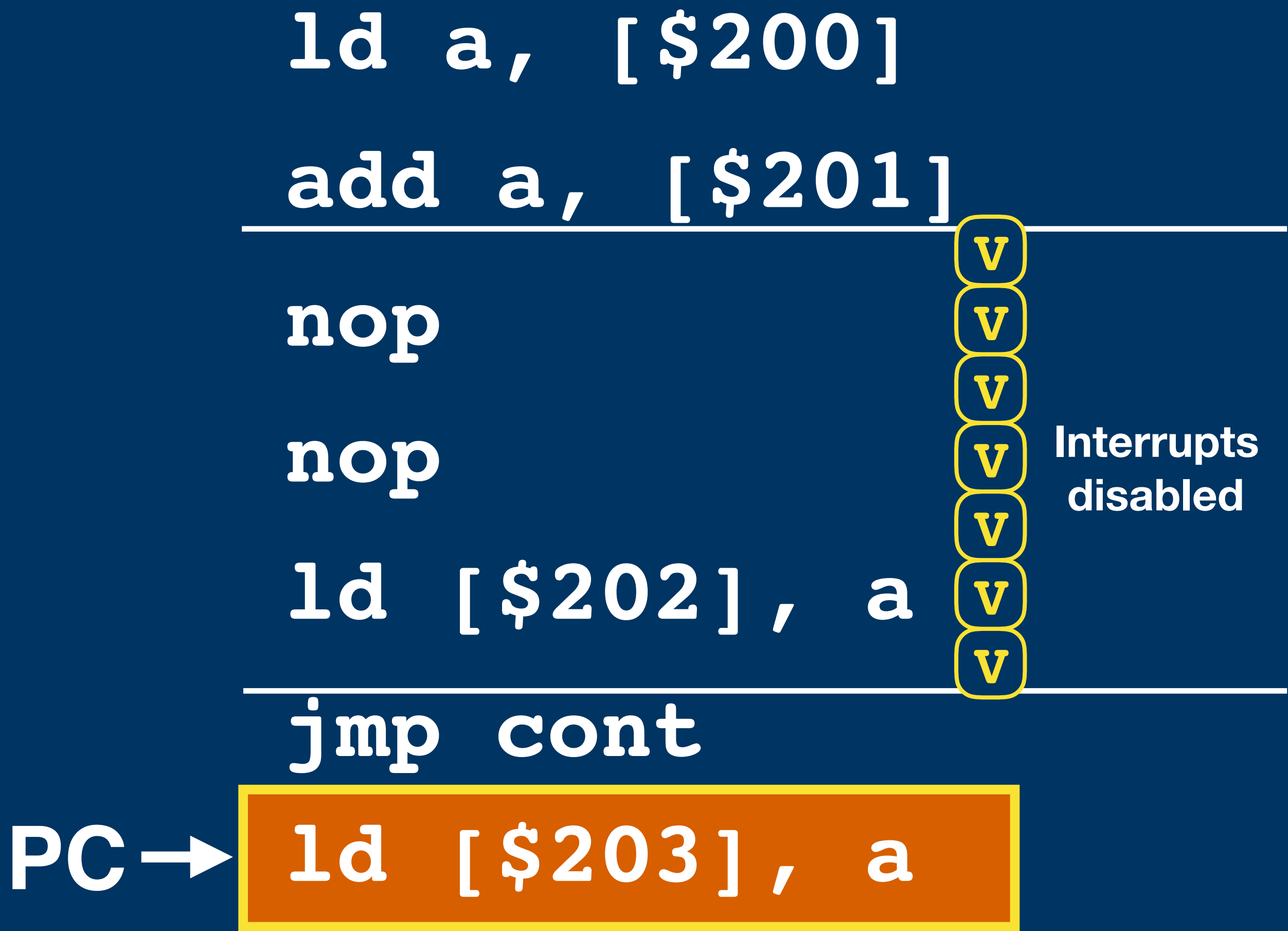
A	0007	000
B	0F00	001
LR	05AA	002
EB	0000	003
FB	0000	004
PC	0421	005
BB	0000	006
0	0000	007

PC → `ld a, [$200]`  
`add a, [$201]`  
`nop`  
`nop`  
`ld [$202], a`  
`jmp cont`  
`ld [$203], a`

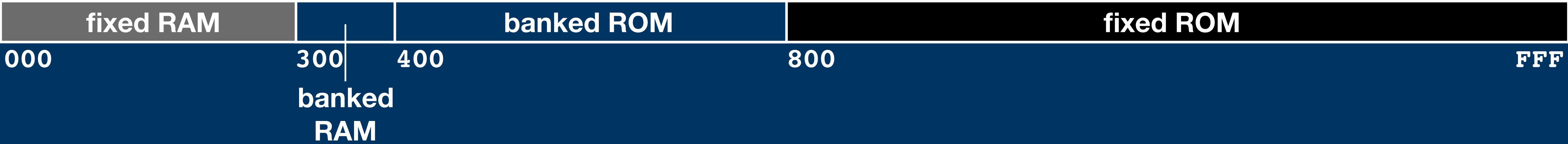


# Interrupts & Overflow

A	0000	000
B	0F00	001
LR	05AA	002
EB	0000	003
FB	0000	004
PC	0421	005
BB	0000	006
0	0000	007

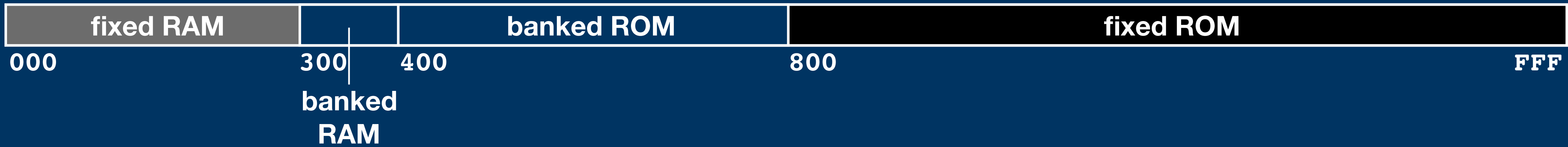


Interrupts



# Interrupts

- INT 0 →
- INT 1 →
- INT 2 →
- INT 3 →
- INT 4 →
- INT 5 →
- INT 6 →
- INT 7 →
- INT 8 →
- INT 9 →
- INT 10 →

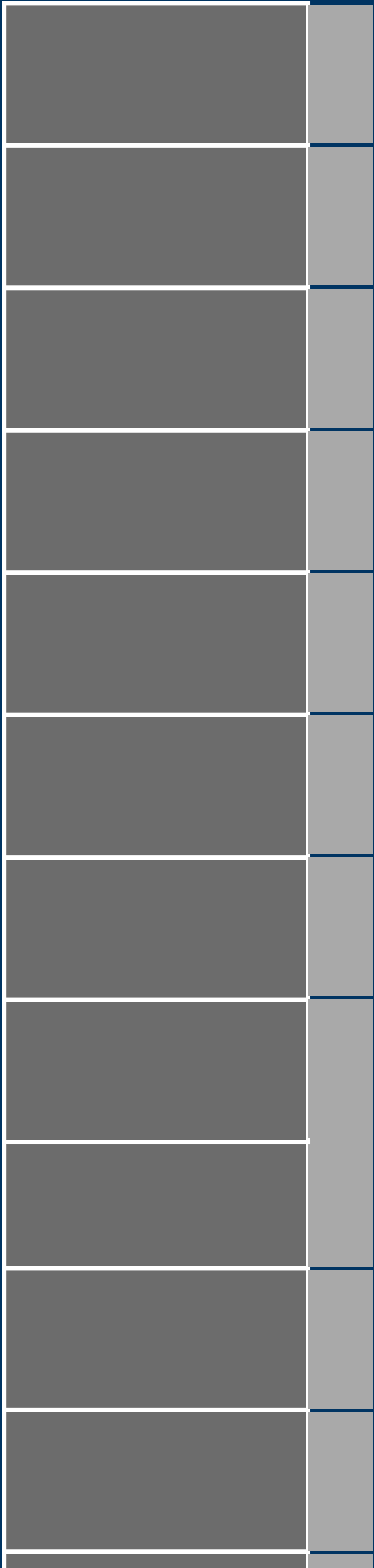


Interrupts

INT 0 →

INT 1 →

INT 2 →



Interrupts

INT 0 →	0004	800	cli
	382c	801	ld a, #\$1443
	5c06	802	xchg a, bbank
	15b7	803	jmp \$5b7
PC →	5409	804	xchg ab, ab'
	0006	805	ld ab, #\$435,\$3c46
	382e	806	
	5406	807	callfbb
INT 2 →	5409	808	xchg ab, ab'
	0006	809	ld ab, [\$2bc]
		810	



Interrupts

INT 0 →	0004	800	cli
	382c	801	ld a, #\$1443
	5c06	802	xchg a, bbank
	15b7	803	jmp \$5b7
INT 1 →	5409	804	xchg ab, ab'
PC →	0006	805	ld ab, #\$435,\$3c46
	382e	806	
	5406	807	callfbb
INT 2 →	5409	808	xchg ab, ab'
	0006	809	ld ab, [\$2bc]
		810	

Interrupts

INT 0 →	0004	800	cli
	382c	801	ld a, #\$1443
	5c06	802	xchg a, bbank
	15b7	803	jmp \$5b7
INT 1 →	5409	804	xchg ab, ab'
	0006	805	ld ab, #\$435,\$3c46
	382e	806	
PC →	5406	807	callfbb
INT 2 →	5409	808	xchg ab, ab'
	0006	809	ld ab, [\$2bc]
		810	

Interrupts

INT 0 →	0004	800	cli
	382c	801	ld a, #\$1443
	5c06	802	xchg a, bbank
	15b7	803	jmp \$5b7
INT 1 →	5409	804	xchg ab, ab'
	0006	805	ld ab, #\$435,\$3c46
	382e	806	
	5406	807	callfbb

Interrupts

RESET →	0004	800	cli
	382c	801	ld a, #\$1443
	5c06	802	xchg a, bbank
	15b7	803	jmp \$5b7
INT 1 →	5409	804	xchg ab, ab'
	0006	805	ld ab, #\$435,\$3c46
	382e	806	
	5406	807	callfbb

# Interrupts



# Interrupts

Int Return	iret		RESUME	
	0017			2
	(enable int) PC ← PC' IR ← IR'			

# Interrupts

Int Return	iret		RESUME	
	0017			2
	(enable int) PC ← PC' IR ← IR'			

Interrupt	int k		EDRUPT k	
	0006, 0E00 + k			4
	(it's complicated)			

# Interrupts

Int Return	iret		RESUME	
	0017			2
	(enable int) PC ← PC' IR ← IR'			

Int Off	cli		INHINT	
	0004			1
	(disable int)			

Interrupt	int k		EDRUPT k	
	0006, 0E00 + k			4
	(it's complicated)			

Int On	sti		RELINT	
	0003			1
	(enable int)			

Watchdog

0

Registers

8

Shadow

\$10

Editing

\$14

Counters

\$31

Watchdog

\$10

\$14

\$31

Shadow

Editing

Counters



Watchdog

Shadow

\$10

Editing

\$14

Counters

\$31

Watchdog

0000

037

access every 0.64 seconds  
otherwise RESET

# Load/Store

Load Cpl	ldc a, [k]	CS k	Load	ld a, [k]	CA k	Xchg	xchg a, [k]	XCH k			
	4000+k			2	3000+k		2	5C00+k		2	
	A ← -M[k]			A ← M[k]			A ↔ M[k]				
Load Dbl Cpl	ldc ab, [k]	DCS k	Load Dbl	ldc ab, [k]	DC k	Xchg Dbl	xchg ab, [k]	DXCH k			
	0006, 4001+k			4	0006, 3001+k		4	5201+k		3	
	A ← -M[k] B ← -M[k+1]			A ← M[k] B ← M[k+1]			A ↔ M[k] B ↔ M[k+1]				

Store	ld [k], a	TS k
	5800+k	2
	M[k] ← A	

In	in a, [kc]	READ kc	Out	out [kc], a	WRITE kc
	0006, 0000+kc	3		0006, 0200+kc	3
	A ← IO[kc]			IO[kc] ← A	
In AND	in& a, [kc]	RAND kc	Out AND	out& [kc], a	WAND kc
	0006, 0400+kc	3		0006, 0600+kc	3
	A ← A & IO[kc]			A & IO[kc] ← A & IO[kc]	
In OR	in  a, [kc]	ROR kc	Out OR	out  [kc], a	WOR kc
	0006, 0800+kc	3		0006, 0A00+kc	3
	A ← A   IO[kc]			A, IO[kc] ← A   IO[kc]	
In XOR	in^ a, [kc]	RXOR k	Out XOR	out^ [kc], a	WXOR kc
	0006, 0C00+kc	3		0006, 0E00+kc	3
	A ← A ^ IO[kc]			A ^ IO[kc] ← A ^ IO[kc]	

Add	add a, [k]	AD k	Subtract	sub a, [k]	SU k
	6000+k	2		0006, 6000+k	3
	A ← A + M[k]			A ← A - M[k]	
Add Strg	add [k]	ADS k	2's Sub	sub2 a, [k]	MSU k
	2C00+k	2		0006, 2000+k	3
	A, M[k] ← A + M[k]			A ← A - <sub>2</sub> M[k]	
Add Double	add [k], ab	DAS k	Increment	inc [k]	INCR k
	2001+k	3		2800+k	2
	M[k] ← M[k] + 1			M[k] ← M[k] + 1	
Augment	aug [k]	AUG k	Diminish	dim [k]	DIM k
	0006, 2800+k	3		0006, 2C00+k	3
	if M[k] ≥ +0: M[k] ← M[k] + 1 if M[k] ≤ -0: M[k] ← M[k] - 1			if M[k] > +0: M[k] ← M[k] - 1 if M[k] < -0: M[k] ← M[k] + 1	
Multiply	mul [k]	MP k	Divide	div [k]	DV k
	0006, 7000+k	4		0006, 1000+k	7
	A, B ← A × M[k]			A ← A, B ÷ M[k] B ← A, B % M[k]	

AND	and a, [k]	MASK k
	7000+k	2
	A ← A & M[k]	

# Control Flow

Jump	jmp k	TCF k
	1000+k	1
	PC ← k	
Jump = 0	jz k	BZF k
	0006, 1000+k	2-3
	if A=±0: PC ← k	
Jump ≤ 0	jlez k	BZMF k
	0006, 6000+k	2-3
	if A ≤ ±0: PC ← k	
Count, Compare and Skip	cc [k]	CC k
	1000+k	2
	PC ← DIM(M[k]) if M[k] > 0: PC ← PC + 1 if M[k] < -0: PC ← PC + 2 if M[k] = -0: PC ← PC + 3	
Call	call k	TC k
	0000+k	1
	LR ← PC PC ← k	

Interrupt	int k	EDRUPT k
	0006, 0E00+k	4
	(disable int) PC' ← PC IR ← IIR'	
Int On	sti	RELINT
	0003	1
	(enable int)	
Int Return	iret	RESUME
	0017	2
	(enable int) PC ← PC' IIR ← IIR'	
Int Off	cli	INHINT
	0004	1
	(disable int)	

# Interrupts

# Instruction Encoding

## Instruction Encoding

3100	400	ld a, [\$100]
6101	401	add a, [\$101]
1405	402	jmp \$405

# Instruction Encoding

3100

400

ld a, [\$100]



## Instruction Encoding

0	1	1	0	0	0	1	0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

**ld a, [\$100]**

# Instruction Encoding

Opcode

3  
0 1 1 0 0 0 1 0 0 0 0 0 0 0 0

ld a, [\$100]

# Instruction Encoding



ld a, [\$100]

# Instruction Encoding



# Instruction Encoding

[illegible]



# Address Encoding

RAM

ROM

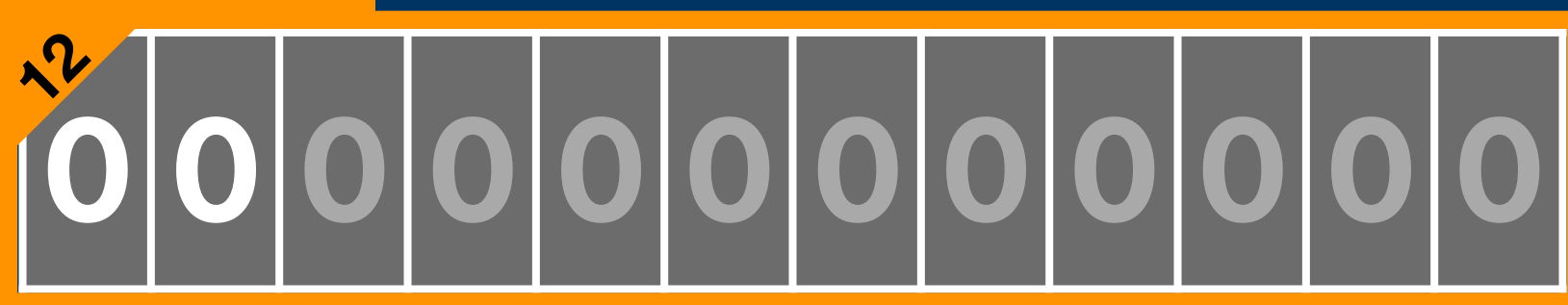
000

3FF 400

FFF

## Address Encoding

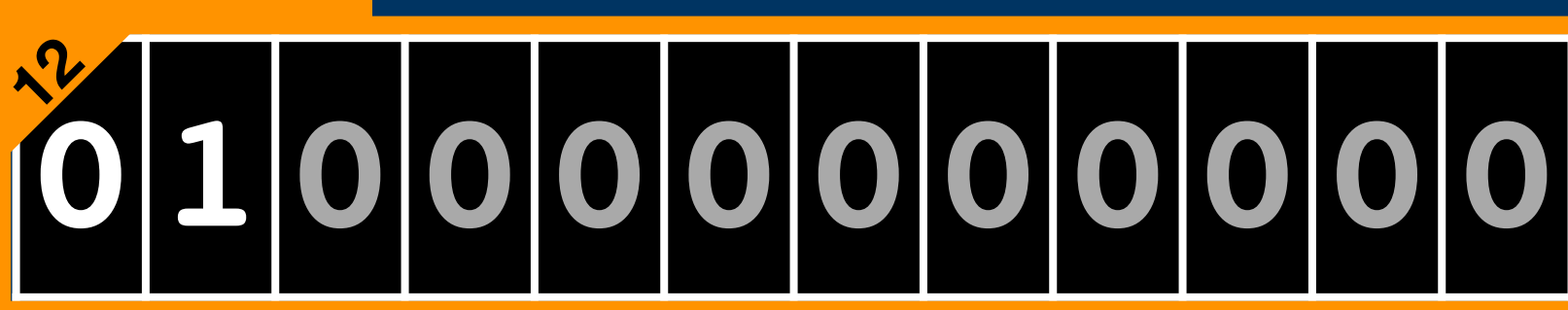
Address



Address



Address



Address



0	0	x	x	x	x	x	x	x	x	x	x	x	RAM
0	1	x	x	x	x	x	x	x	x	x	x	x	
1	0	x	x	x	x	x	x	x	x	x	x	x	ROM
1	1	x	x	x	x	x	x	x	x	x	x	x	

RAM

ROM

000

3FF 400

FFF

Instruction Encoding

Opcode			Address												
0	0	0													call k
0	0	1													ccs [k]
0	1	0													inc [k]
0	1	1													ld a, [k]
1	0	0													ldc a, [k]
1	0	1													ld [k], a
1	1	0													add a, [k]
1	1	1													and a, [k]

RAM

RAM/ROM

ROM

Instruction Encoding

Opcode			Address												
0	0	0													call k
0	0	1													ccs [k]
0	1	0													inc [k]
0	1	1													ld a, [k]
1	0	0													ldc a, [k]
1	0	1	0	0											ld [k], a
1	1	0													add a, [k]
1	1	1													and a, [k]

RAM

RAM/ROM

ROM

Instruction Encoding

Opcode			Address												
0	0	0													call k
0	0	1													ccs [k]
0	1	0													inc [k]
0	1	1													ld a, [k]
1	0	0													ldc a, [k]

1	0	1	0	0											ld [k], a
---	---	---	---	---	--	--	--	--	--	--	--	--	--	--	-----------

1	1	0													add a, [k]
1	1	1													and a, [k]

RAM

RAM/ROM

ROM



Instruction Encoding

Opcode			Address												
0	0	0													call k
0	0	1													ccs [k]
0	1	0													inc [k]
0	1	1													ld a, [k]
1	0	0													ldc a, [k]

1	0	1	0	0											ld [k], a
---	---	---	---	---	--	--	--	--	--	--	--	--	--	--	-----------

1	1	0													add a, [k]
1	1	1													and a, [k]

RAM

RAM/ROM

ROM

Instruction Encoding

Opcode			Address												
0	0	0													call k
0	0	1													ccs [k]
0	1	0													inc [k]
0	1	1													ld a, [k]
1	0	0													ldc a, [k]
1	0	1	0	0											INDEX
1	0	1	0	1											xchg ab, [k]
1	0	1	1	0											ld [k], a
1	0	1	1	1											xchg a, [k]
1	1	0													add a, [k]
1	1	1													and a, [k]

RAM

RAM/ROM

ROM

Instruction Encoding

Opcode			Address												
0	0	0													call k
0	0	1													ccs [k]
0	1	0													inc [k]
0	1	1													ld a, [k]
1	0	0													ldc a, [k]
1	0	1	0	0											INDEX
1	0	1	0	1											xchg ab, [k]
1	0	1	1	0											ld [k], a
1	0	1	1	1											xchg a, [k]
1	1	0													add a, [k]
1	1	1													and a, [k]

RAM

RAM/ROM

ROM

Instruction Encoding

Opcode			Address												
0	0	0													call k
0	0	1													ccs [k]

0	1	0	0	0											inc [k]
---	---	---	---	---	--	--	--	--	--	--	--	--	--	--	---------

0	1	1													ld a, [k]
1	0	0													ldc a, [k]
1	0	1	0	0											INDEX
1	0	1	0	1											xchg ab, [k]
1	0	1	1	0											ld [k], a
1	0	1	1	1											xchg a, [k]
1	1	0													add a, [k]
1	1	1													and a, [k]

RAM

RAM/ROM

ROM

Instruction Encoding

Opcode			Address												
0	0	0													call k
0	0	1													ccs [k]

0	1	0	0	0											inc [k]
---	---	---	---	---	--	--	--	--	--	--	--	--	--	--	---------

0	1	1													ld a, [k]
1	0	0													ldc a, [k]
1	0	1	0	0											INDEX
1	0	1	0	1											xchg ab, [k]
1	0	1	1	0											ld [k], a
1	0	1	1	1											xchg a, [k]
1	1	0													add a, [k]
1	1	1													and a, [k]

RAM

RAM/ROM

ROM



Instruction Encoding

Opcode			Address												
0	0	0													call k
0	0	1													ccs [k]
0	1	0	0	0											add [k], ab
0	1	0	0	1											xchg b, [k]
0	1	0	1	0											inc [k]
0	1	0	1	1											add [k], a
0	1	1													ld a, [k]
1	0	0													ldc a, [k]
1	0	1	0	0											INDEX
1	0	1	0	1											xchg ab, [k]
1	0	1	1	0											ld [k], a
1	0	1	1	1											xchg a, [k]
1	1	0													add a, [k]
1	1	1													and a, [k]

RAM

RAM/ROM

ROM

Instruction Encoding

Opcode			Address												
0	0	0													call k
0	0	1													ccs [k]
0	1	0	0	0											add [k], ab
0	1	0	0	1											xchg b, [k]
0	1	0	1	0											inc [k]
0	1	0	1	1											add [k], a
0	1	1													ld a, [k]
1	0	0													ldc a, [k]
1	0	1	0	0											INDEX
1	0	1	0	1											xchg ab, [k]
1	0	1	1	0											ld [k], a
1	0	1	1	1											xchg a, [k]
1	1	0													add a, [k]
1	1	1													and a, [k]

RAM

RAM/ROM

ROM

Instruction Encoding

Opcode			Address												
0	0	0													call k
0	0	1	0	0											ccs [k]

0	1	0	0	0											add [k], ab
0	1	0	0	1											xchg b, [k]
0	1	0	1	0											inc [k]
0	1	0	1	1											add [k], a
0	1	1													ld a, [k]
1	0	0													ldc a, [k]
1	0	1	0	0											INDEX
1	0	1	0	1											xchg ab, [k]
1	0	1	1	0											ld [k], a
1	0	1	1	1											xchg a, [k]
1	1	0													add a, [k]
1	1	1													and a, [k]

RAM

RAM/ROM

ROM

Instruction Encoding

Opcode			Address												
0	0	0													call k
0	0	1	0	0											ccs [k]

0	1	0	0	0											add [k], ab
0	1	0	0	1											xchg b, [k]
0	1	0	1	0											inc [k]
0	1	0	1	1											add [k], a
0	1	1													ld a, [k]
1	0	0													ldc a, [k]
1	0	1	0	0											INDEX
1	0	1	0	1											xchg ab, [k]
1	0	1	1	0											ld [k], a
1	0	1	1	1											xchg a, [k]
1	1	0													add a, [k]
1	1	1													and a, [k]

RAM

RAM/ROM

ROM

Instruction Encoding

Opcode			Address												
0	0	0													call k
0	0	1	0	0											ccs [k]
0	0	1													jmp
0	1	0	0	0											add [k], ab
0	1	0	0	1											xchg b, [k]
0	1	0	1	0											inc [k]
0	1	0	1	1											add [k], a
0	1	1													ld a, [k]
1	0	0													ldc a, [k]
1	0	1	0	0											INDEX
1	0	1	0	1											xchg ab, [k]
1	0	1	1	0											ld [k], a
1	0	1	1	1											xchg a, [k]
1	1	0													add a, [k]
1	1	1													and a, [k]

\* call can be used to work around the ROM limitation of jmp

\*

RAM

RAM/ROM

ROM



# Instruction Encoding



# Instruction Encoding

A	0007	000
B	0F00	001
LR	05AA	002
EB	0000	003
FB	0000	004
PC	0421	005
BB	0000	006
0	0000	007

Opcode			Address												
0	0	0													call k
0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	sti
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	cli
0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	EXTEND

# Instruction Encoding

[illegible]

# Instruction Encoding

[illegible]

# Instruction Encoding





# Instruction Encoding

EXTEND



I/O

RAM

RAM/ROM

ROM

Instruction Encoding

EXTEND

Opcode						Address									
0	0	0	0	0	0										in [kc]
0	0	0	0	0	1										out [kc]
0	0	0	0	1	0										in& [kc]
0	0	0	0	1	1										out& [kc]
0	0	0	1	0	0										in  [kc]
0	0	0	1	0	1										out  [kc]
0	0	0	1	1	0										in^ [kc]
0	0	0	1	1	1										edrapt k
0	0	1	0	0											div [k]
0	0	1													jz k
0	1	0	0	0											sub2 a, [k]
0	1	0	0	1											xchg lr, [k]
0	1	0	1	0											aug [k]
0	1	0	1	1											dim [k]
0	1	1													ld ab, [k]
1	0	0													ldc ab, [k]
1	0	1													INDEX
1	1	0	0	0											sub a, [k]
1	1	0													jlez k
1	1	1													mul [k]

I/O

RAM

RAM/ROM

ROM

# Instruction Encoding

A	0007	000
B	0F00	001
LR	05AA	002
EB	0000	003
FB	0000	004
PC	0421	005
BB	0000	006
0	0000	007

Opcode			Address												
0	0	0													call k
0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	sti
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	cli
0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	EXTEND

# Instruction Encoding

A	0007	000
B	0F00	001
LR	05AA	002
EB	0000	003
FB	0000	004
PC	0421	005
BB	0000	006
0	0000	007

Opcode			Address												
0	0	0													call k
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	ret
0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	sti
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	cli
0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	EXTEND

Return

A	0007	000
B	0F00	001
LR	0421	002
EB	0000	003
FB	0000	004
PC	0E81	005
BB	0000	006
0	0000	007

ret

373D	E
3100	41F
0E80	420
3103	421
EDFF	42
5004	F
5880	E80
0002	E81
	E

PC →

Return	ret	RETURN
	call 2	alias 1
	LR ← PC   PC ← 2	

ld a, [\$100]

call \$e80

ld a, [\$103]

ld [\$80], a

ret



Return

ret

Return

ret

RETURN

call 2

alias

1

LR ← PC

PC ← 2

A

0007

000

B

0F00

001

LR

0421

002

EB

0000

003

FB

0000

004

PC

0E81

005

BB

0000

006

0

0000

007

3100

41F

0E80

420

3103

421

5880

E80

0002

E81

ld a, [\$100]

call \$e80

ld a, [\$103]

ld [\$80], a

call 2

PC →

Return

ret

Return

ret

RETURN

call 2

alias

1

LR ← PC | PC ← 2

A	0007	000
B	0F00	001
LR	0421	002
EB	0000	003
FB	0000	004
PC	0002	005
BB	0000	006
0	0000	007

373D	E
3100	41F
0E80	420
3103	421
EDFF	42
5004	F
5880	E80
0002	E81
	E

ld a, [\$100]

call \$e80

ld a, [\$103]

ld [\$80], a

call 2

Return

ret

Return

ret

RETURN

call 2

alias

1

LR ← PC

PC ← 2

Opcode

Address

0421

A	0007	000
B	0F00	001
LR	0421	002
EB	0000	003
FB	0000	004
PC	0002	005
BB	0000	006
0	0000	007

373D	E
3100	4
0E80	0
5880	E80
0004	F
0002	E81
	E

ld a, [\$100]

call \$e80

ld a, [\$103]

ld [\$80], a

call 2

Return

ret

Return

ret

RETURN

call 2

alias

1

LR ← PC

PC ← 2

Opcode

Address

0 0 0 0 1 0 0 0 0 1 0 0 0 0 1

A

0007

B

0F00

LR

0421

EB

0000

FB

0000

PC

0002

BB

0000

0

0000

373D

3100

0E80

3103

EDFF

3804

5880

0002

E

4

0

4

2

F

E

E

ld a, [\$100]

call \$e80

ld a, [\$103]

ld [\$80], a

call 2

Return

ret

Return	ret		RETURN	
	call 2			alias 1
	LR ← PC		PC ← 2	

A 0007

B 0F00

LR 0421

EB 0000

FB 0000

PC 0002

BB 0000

0 0000

Opcode

000010000100001

Address

call k

373D

3100

0E80

3103

EDFF

3804

5880

0002

E

4

0

421

42

F

E80

E81

E

ld a, [\$100]

call \$e80

ld a, [\$103]

ld [\$80], a

call 2



Return

ret

Return	ret		RETURN	
	call 2			alias 1
	LR ← PC		PC ← 2	

A

0007

B

0F00

Opcode

Address

000010000100001

call \$421

LR

0421

EB

0000

FB

0000

PC

0002

BB

0000

0

0000

373D

3100

0E00

3103

EDFF

5004

5880

0002

E

4

0

421

42

F

E80

E81

ld a, [\$100]

call \$E00

ld a, [\$103]

ld [\$80], a

call 2

Return

ret

Return

ret

RETURN

call 2

alias

1

LR ← PC

PC ← 2

A	0007	000
B	0F00	001
LR	0421	002
EB	0000	003
FB	0000	004
PC	0421	005
BB	0000	006
0	0000	007

\*

PC →

3100	41F
0E80	420
3103	421
5880	E80
0002	E81

ld a, [\$100]

call \$e80

ld a, [\$103]

ld [\$80], a

call 2

\* because it's a "call",  
LR will also be updated  
after every instruction  
(not shown here)

INDEX

## INDEX

1

**O**

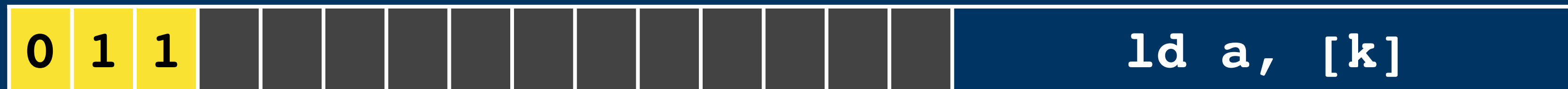
1

**O**

**O**

## INDEX

INDEX



ld a, [\$700+[\$80]]



INDEX

1	0	1	0	0	0	0	1	0	0	0	0	0	0	0	INDEX [\$80]
0	1	1	0	1	1	1	0	0	0	0	0	0	0	0	ld a, [\$700]

ld a, [\$700+[\$80]]

INDEX

Indexing	INDEX k		INDEX k	
	6000+k			2
	IR ← IR + M[k]			

INDEX

Indexing	INDEX k	
	6000+k	
	IR ← IR + M[k]	

PC →

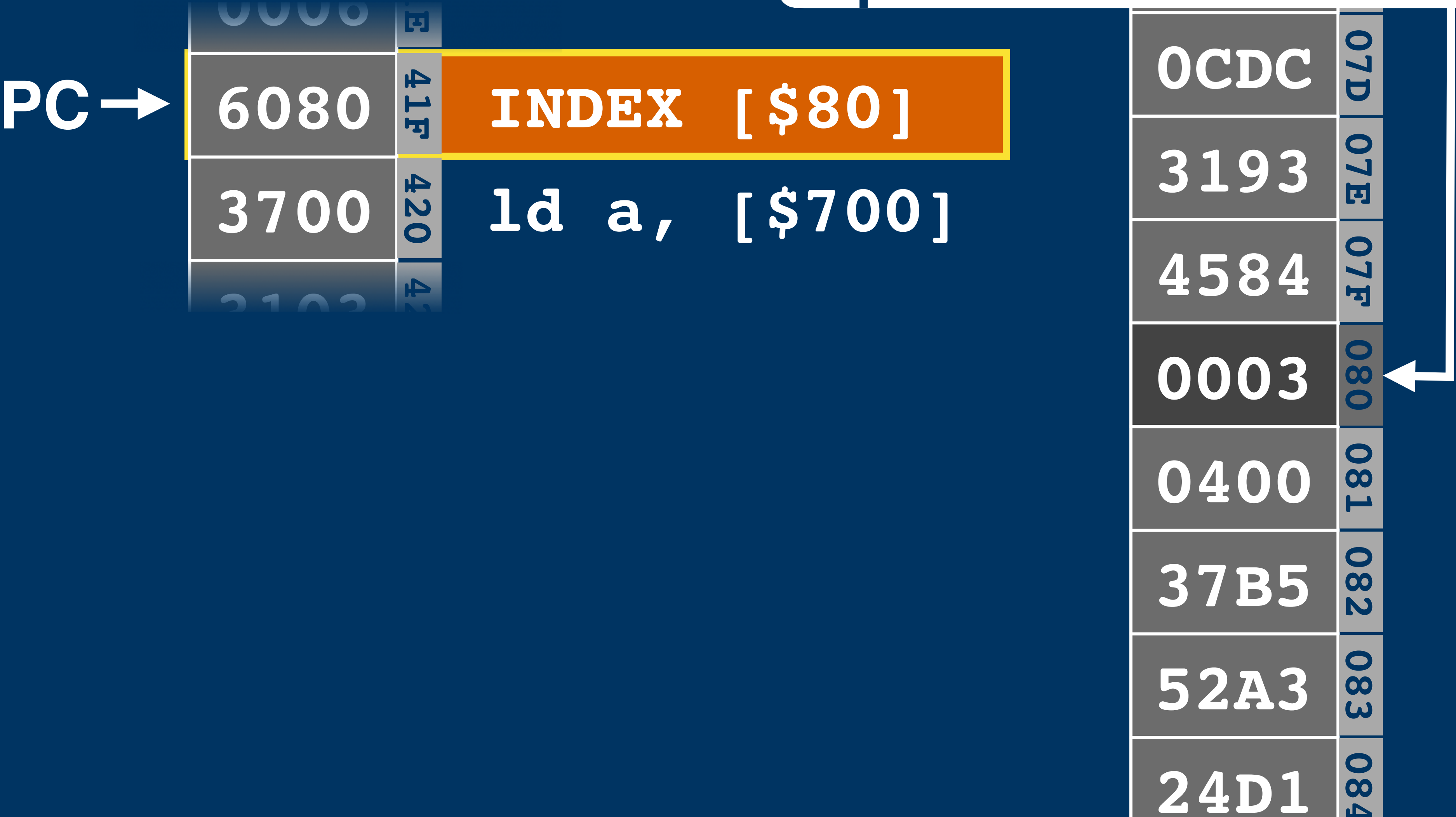
6080	41F
3700	420
	42

INDEX [\$80]

ld a, [\$700]

INDEX

Indexing	INDEX k	INDEX k	
	6000+k		2
	IR ← IR + M[k]		



INDEX

Indexing

INDEX k

INDEX k

6000+k

2

IR ← IR + M[k]

PC →

6080

41F

INDEX [\$80]

3700

420

ld a, [\$700]

3700

+

0003

IR

3703

0CDC

07D

3193

07E

4584

07F

0003

080

0400

081

37B5

082

52A3

083

24D1

084



INDEX

Indexing	INDEX k	INDEX k	
	6000+k		2
	IR ← IR + M[k]		

PC →  
IR

0000	E
6080	41F
3703	420
3103	42

INDEX [\$80]

ld a, [\$700]

0CDC	07D
3193	07E
4584	07F
0003	080
0400	081
37B5	082
52A3	083
24D1	084



INDEX

Indexing

INDEX k

INDEX k

6000+k

2

IR ← IR + M[k]

PC →

6080

INDEX [\$80]

IR

3703

ld a, [\$703]

0CDC

07D

3193

07E

4584

07F

0003

080

0400

081

37B5

082

52A3

083

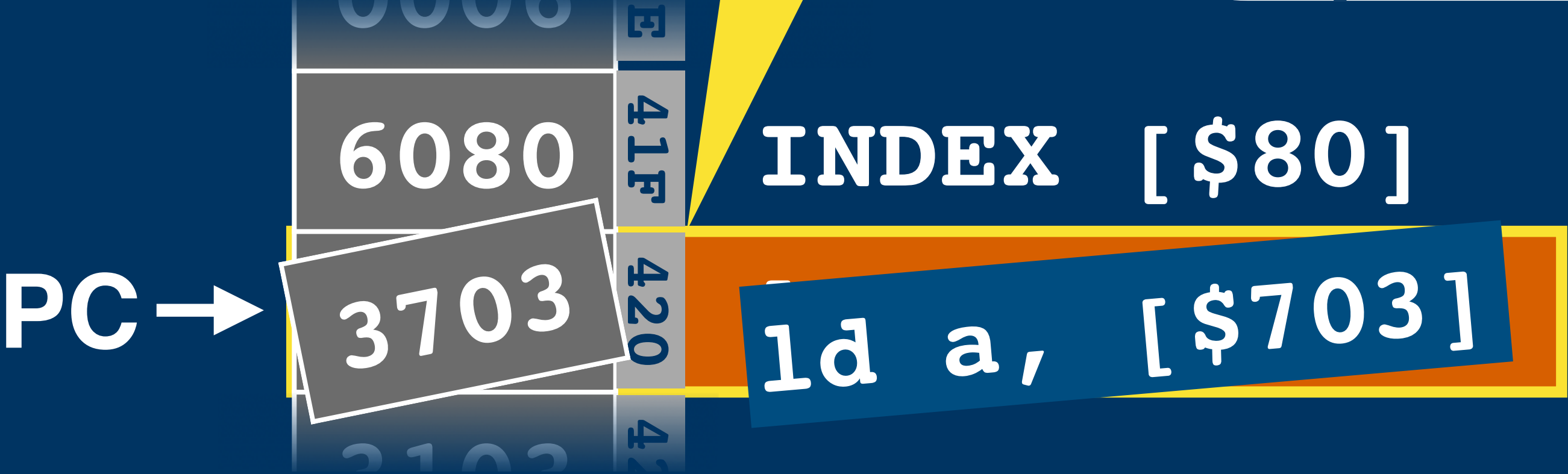
24D1

084

INDEX

Interrupt

Indexing	INDEX k	INDEX k	
	6000+k		2
	IR ← IR + M[k]		



0CDC	07D
3193	07E
4584	07F
0003	080
0400	081
37B5	082
52A3	083
24D1	084

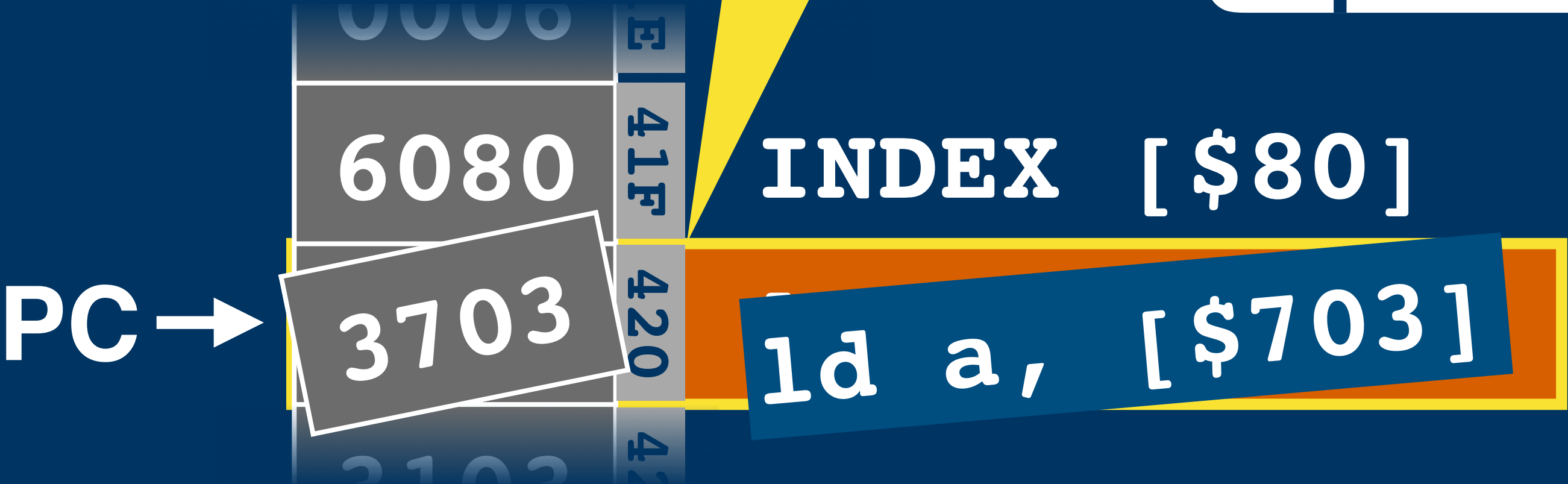
IR 3703

IR' 0000

INDEX

Interrupt

Indexing	INDEX k	INDEX k	
	6000+k		2
	IR ← IR + M[k]		



0CDC	07D
3193	07E
4584	07F
0003	080
0400	081
37B5	082
52A3	083
24D1	084

IR 3703

IR' 3703

# Instruction Encoding

A'	0000	008
B'	0000	009
LR'	0000	00A
	0000	00B
	0000	00C
PC'	0000	00D
BB'	0000	00E
IR'	0000	00F





# Instruction Encoding

A'	0000	008
B'	0000	009
LR'	0000	00A
	0000	00B
	0000	00C
PC'	0000	00D
BB'	0000	00E
IR'	0000	00F

Opcode					Address											
1	0	1	0	0											INDEX	
1	0	1	0	0	0	0	0	0	0	0	1	1	1	1	iret	

## Quirks

# Quirks

## One's Complement



## Interrupts & Overflow

A	0000
B	0F00
LR	05AA
EB	0000
FB	0000
PC	0421
BB	0000
0	0000

PC → **ld a, [\$200]**

**add a, [\$201]**

**nop**

**nop**

**ld [\$202], a**

**jmp cont**

**ld [\$203], a**



## Overflow Handling

A	0001
B	0F00
LR	05AA
EB	0000
FB	0000
PC	0421
BB	0000
0	0000



PC → **ld a, [\$200]**

**add a, [\$201]**

**ld [\$202], a**

**jmp cont**

**ld [\$203], a**

7FFF
0001
0000
0001

Double Word Result

## Conditional Jumps

ccs [k]	CCS k
1000+k	2
$A \leftarrow \text{DABS}(M[k])$ if $M[k] > +0$ : $PC \leftarrow PC$ if $M[k] = +0$ : $PC \leftarrow PC+1$ if $M[k] < -0$ : $PC \leftarrow PC+2$ if $M[k] = -0$ : $PC \leftarrow PC+3$	

1200	41F	ccs [\$200]
0430	420	jmp \$430
0440	421	jmp \$440
0450	422	jmp \$450
3080	62E	ld a, [\$80]



## Editing

ROR 101011100100101

SHR 001011100100101

ROL 101110010010110

SHR8 010111001001011

## I/O Channels

B	000
LR	001
	002
	003
	004
	005
	006
	007
	008
	009
	010
	011
	012
	013
	014
	015
	016
	017
	018
	019
	020
	021
	022
	023
	024
	025
	026
	027
	028
	029
	030
	031
	032
	033
	034
	035
	036
	037
	038
	039
	040
	041
	042
	043
	044
	045
	046
	047
	048
	049
	050
	051
	052
	053
	054
	055
	056
	057
	058
	059
	060
	061
	062
	063
	064
	065
	066
	067
	068
	069
	070
	071
	072
	073
	074
	075
	076
	077
	078
	079
	080
	081
	082
	083
	084
	085
	086
	087
	088
	089
	090
	091
	092
	093
	094
	095
	096
	097
	098
	099
	100
	101
	102
	103
	104
	105
	106
	107
	108
	109
	110
	111
	112
	113
	114
	115
	116
	117
	118
	119
	120
	121
	122
	123
	124
	125
	126
	127
	128
	129
	130
	131
	132
	133
	134
	135
	136
	137
	138
	139
	140
	141
	142
	143
	144
	145
	146
	147
	148
	149
	150
	151
	152
	153
	154
	155
	156
	157
	158
	159
	160
	161
	162
	163
	164
	165
	166
	167
	168
	169
	170
	171
	172
	173
	174
	175
	176
	177
	178
	179
	180
	181
	182
	183
	184
	185
	186
	187
	188
	189
	190
	191
	192
	193
	194
	195
	196
	197
	198
	199
	200
	201
	202
	203
	204
	205
	206
	207
	208
	209
	210
	211
	212
	213
	214
	215
	216
	217
	218
	219
	220
	221
	222
	223
	224
	225
	226
	227
	228
	229
	230
	231
	232
	233
	234
	235
	236
	237
	238
	239
	240
	241
	242
	243
	244
	245
	246
	247
	248
	249
	250
	251
	252
	253
	254
	255

In	in a, [kc]	READ kc
	0006, 0000+kc	2
	$A \leftarrow \text{IO}[kc]$	
Out	out [kc], a	WRITE kc
	0006, 0200+kc	2
	$\text{IO}[kc] \leftarrow A$	
In AND	in& a, [kc]	RAND kc
	0006, 0400+kc	2
	$A \leftarrow A \& \text{IO}[kc]$	
In OR	in  a, [kc]	ROR kc
	0006, 0800+kc	2
	$A \leftarrow A   \text{IO}[kc]$	
In XOR	in^ a, [kc]	RXOR kc
	0006, 0C00+kc	2
	$A \leftarrow A \wedge \text{IO}[kc]$	

## INDEX

## Interrupt

INDEX k	INDEX k
6000+k	2
$IR \leftarrow IR + M[k]$	

PC → **INDEX [\$80]**

**ld a, [\$702]**

IR	3700
	+
IR	3702

0CDC	07D
3193	07E
4584	07F
0002	080
0400	081
37B5	082
52A3	083
24D1	084

## Indexing

**ld a, [\$700+\$80]**

A	0006
B	0F00
LR	05AA
EB	0000
FB	0000
PC	0421
BB	0000
0	0000

0000	6FE
0000	6FF
0000	700
0003	701
0006	702
0009	703
000C	704
000F	705
0012	706

0700	07C
0002	07D
0702	07E
133A	07F
33DF	080
1093	081
23C9	082
0002	083
291C	084
20F7	085
100F	086
45B3	087

# Quirks

## One's Complement

# One's Complement

## Interrupts & Overflow

A	0000	000
B	0F00	001
LR	05AA	002
EB	0000	003
FB	0000	004
PC	0421	005
BB	0000	006
0	0000	007

PC → **ld a, [\$200]**

**add a, [\$201]**

**nop**

**nop**

**ld [\$202], a**

**jmp cont**

**ld [\$203], a**

V  
V  
V  
V  
V  
V  
V  
V  
Interrupts disabled

## Overflow Handling

A	0001
B	0F00
LR	05AA
EB	0000
FB	0000
PC	0421
BB	0000
0	0000

+ 0000  
- 0000 V

PC → **ld a, [\$200]**

**add a, [\$201]**

**ld [\$202], a**

**jmp cont**

**ld [\$203], a**

7FFF	200
0001	201
0000	202
0001	203

Double Word Result

## Conditional Jumps

Count, Compare and Skip	ccs [k]	CCS k
	1000+k	2
	$A \leftarrow DABS(M[k])$	
	if $M[k] > +0$ :	
	PC $\leftarrow$ PC	
	if $M[k] = +0$ :	
	PC $\leftarrow$ PC+1	
	if $M[k] < -0$ :	
	PC $\leftarrow$ PC+2	
	if $M[k] = -0$ :	
	PC $\leftarrow$ PC+3	

1200	41F	ccs [\$200]
0430	420	jmp \$430
0440	421	jmp \$440
0450	422	jmp \$450
3080	62E	ld a, [\$80]

>+0  
=+0  
<-0  
=-0

## Editing

ROR 101011100100101

SHR 001011100100101

ROL 101110010010110

SHR8 010111001001011

## I/O Channels

B	000
LR	001
	002
	003
	004
	005
	006
	007
	008
	009
	010
	011
	012
	013
	014
	015
	016
	017
	018
	019
	020
	021
	022
	023
	024
	025
	026
	027
	028
	029
	030
	031

In	in a, [kc]	READ kc
	0006, 0000+kc	2
	$A \leftarrow IO[kc]$	
Out	out [kc], a	WRITE kc
	0006, 0200+kc	2
	$IO[kc] \leftarrow A$	
In AND	in& a, [kc]	RAND kc
	0006, 0400+kc	2
	$A \leftarrow A \& IO[kc]$	
In OR	in  a, [kc]	ROR kc
	0006, 0800+kc	2
	$A \leftarrow A   IO[kc]$	
In XOR	in^ a, [kc]	RXOR kc
	0006, 0C00+kc	2
	$A \leftarrow A \wedge IO[kc]$	

## INDEX

## Interrupt

INDEX k	INDEX k
6000+k	2
$IR \leftarrow IR + M[k]$	

PC → **INDEX [\$80]**

**ld a, [\$702]**

IR 3700  
+ 0002  
IR 3702

0CDC	07D
3193	07E
4584	07F
0002	080
0400	081
37B5	082
52A3	083
24D1	084

## Indexing

**ld a, [\$700+\$80]**

A	0006
B	0F00
LR	05AA
EB	0000
FB	0000
PC	0421
BB	0000
0	0000

0000	6FE
0000	6FF
0000	700
0003	701
0006	702
0009	703
000C	704
000F	705
0012	706

+ 0700  
0002  
0702

Index at \$80

133A	07C
33DF	07D
1093	07E
23C9	07F
0002	080
291C	081
20F7	082
100F	083
45B3	084

# Quirks

One's Complement

One's Complement

Interrupts & Overflow

A	0000
B	0F00
LR	05AA
EB	0000
FB	0000
PC	0421
BB	0000
0	0000

add a, [\$201]  
nop  
ld [\$202], a  
ld [\$203], a

Interrupts disabled

Overflow Disables Interrupts

Overflow Handling

A	0001
B	0F00
LR	05AA
EB	0000
FB	0000
PC	0421
BB	0000
0	0000

PC → ld a, [\$200]  
add a, [\$201]  
ld [\$202], a  
jmp cont  
ld [\$203], a

7FFF  
0001  
0000  
0001

Double Word Result

Conditional Jumps

ccs [k]	CCS k
1000+k	2

A ← DABS(M[k])  
if M[k]>+0:  
PC ← PC  
if M[k]=+0:  
PC ← PC+1  
if M[k]<-0:  
PC ← PC+2  
if M[k]=-0:  
PC ← PC+3

1200  
0430  
0440  
0450  
3080

ccs [\$200]  
jmp \$430  
jmp \$440  
jmp \$450  
ld a, [\$80]

>+0  
=+0  
<-0  
=-0

Editing

ROR 101011100100101

SHR 001011100100101

ROL 101110010010110

SHR8 010111001001011

I/O Channels

In	in a, [kc]	READ kc
	0006, 0000+kc	2
	A ← IO[kc]	
In AND	in& a, [kc]	RAND kc
	0006, 0400+kc	2
	A ← A & IO[kc]	
In OR	in  a, [kc]	ROR kc
	0006, 0800+kc	2
	A ← A   IO[kc]	
In XOR	in^ a, [kc]	RXOR kc
	0006, 0C00+kc	2
	A ← A ^ IO[kc]	

Out

out [kc], a	WRITE kc
0006, 0200+kc	2
IO[kc] ← A	
out& [kc], a	WAND kc
0006, 0600+kc	2
A, IO[kc] ← A & IO[kc]	
out  [kc], a	WOR kc
0006, 0A00+kc	2
A, IO[kc] ← A   IO[kc]	

INDEX

Interrupt

INDEXING

INDEX k	INDEX k
6000+k	2

IR ← IR + M[k]

PC → 6080  
INDEX [\$80]  
ld a, [\$702]

IR 3700  
+ 0002  
IR 3702

0CDC  
3193  
4584  
0002  
0400  
37B5  
52A3  
24D1

Indexing

ld a, [\$700+\$80]

A 0006  
B 0F00  
LR 05AA  
EB 0000  
FB 0000  
PC 0421  
BB 0000  
0 0000

Table at \$700

Index at \$80

0700  
+ 0002  
0702

0000  
0000  
0003  
0006  
0009  
000C  
000F  
0012

133A  
33DF  
1093  
23C9  
291C  
20F7  
100F  
45B3



Quirks

One's Complement

One's Complement

Interrupts & Overflow

Overflow Disables Interrupts

Overflow Handling

Store Skips On Overflow

Conditional Jumps

Count, Compare and Skip

Editing

ROR 101011100100101

SHR 001011100100101

ROL 101110010010110

SHR8 010111001001011

I/O Channels

B LR

INDEX

Interrupt

Indexing

ld a, [\$700+\$80]

Quirks

One's Complement

One's Complement

Interrupts & Overflow

Overflow Disables Interrupts

Overflow Handling

Store Skips On Overflow

Conditional Jumps

CCS

Editing

ROR 101011100100101

SHR 001011100100101

ROL 101110010010110

SHR8 010111001001011

I/O Channels

B LR

INDEX

Interrupt

Indexing

ld a, [\$700+\$80]



# Quirks

## One's Complement

# One's Complement

## Interrupts & Overflow

A	0000
B	0F00
LR	05AA
EB	0000
FB	0000
PC	0421
BB	0000
0	0000

# Overflow Disables Interrupts

## Overflow Handling

A	0001
B	0F00
LR	05AA
EB	0000
FB	0000
PC	0421
BB	0000
0	0000

# Store Skips On Overflow

## Conditional Jumps

Count, Compare and Skip	ccs [k]	CCS k
	1000+k	2
	$A \leftarrow DABS(M[k])$	
	if $M[k] > +0$ :	
	PC $\leftarrow$ PC	
	if $M[k] = +0$ :	
	PC $\leftarrow$ PC+1	
	if $M[k] < -0$ :	
	PC $\leftarrow$ PC+2	
	if $M[k] = -0$ :	
	PC $\leftarrow$ PC+3	

# CCS

1200	41F	ccs [\$200]
0450	422	jmp \$430
0440	422	jmp \$440
0450	422	jmp \$450
3080	62E	ld a, [\$80]

## Editing

# Shift/Rotate Using Memory

## I/O Channels

In	in a, [kc]	READ kc	Out	out [kc], a	WRITE kc
	0006, 0000+kc	2		0006, 0200+kc	2
	$A \leftarrow IO[kc]$			$IO[kc] \leftarrow A$	
In AND	in& a, [kc]	RAND kc	Out AND	out& [kc], a	WAND kc
	0006, 0400+kc	2		0006, 0600+kc	2
	$A \leftarrow A \& IO[kc]$			$A, IO[kc] \leftarrow A \& IO[kc]$	
In OR	in  a, [kc]	ROR kc	Out OR	out  [kc], a	WOR kc
	0006, 0800+kc	2		0006, 0A00+kc	2
	$A \leftarrow A   IO[kc]$			$A, IO[kc] \leftarrow A   IO[kc]$	
In XOR	in^ a, [kc]	RXOR kc			
	0006, 0C00+kc	2			
	$A \leftarrow A \wedge IO[kc]$				

## INDEX

## Interrupt

Indexing	INDEX k	INDEX k
	6000+k	2
	$IR \leftarrow IR + M[k]$	

PC $\rightarrow$	6080	41F	INDEX [\$80]
	3700	420	ld a, [\$702]
			IR 3700
			IR 3702

## Indexing

A	0006
B	0F00
LR	05AA
EB	0000
FB	0000
PC	0421
BB	0000
0	0000

0000	6FE	0700	133A
0000	6FF	0002	33DF
0000	700	0000	1093
0003	701	0000	23C9
0006	702	0002	0002
0009	703	0000	291C
000C	704	0000	20F7
000F	705	0000	100F
0012	706	0000	45B3

## Quirks

A circular number line on a dark blue background. The line is marked with integers from -8 to 7. The numbers are arranged in a circle, with -8 at the top and 7 at the bottom. The numbers are in a light blue color. Overlaid on the center of the circle is the text "One's Complement" in a large, white, sans-serif font. The text is split across two lines: "One's" on the top line and "Complement" on the bottom line. The text is slightly transparent, allowing the circular number line to be seen through it.

Diagram illustrating the state of a processor during an interrupt event, showing the sequence of instructions and the resulting overflow condition.

**Interrupts & Overflow**

The diagram shows a sequence of instructions and their corresponding register values (A, B, LR, EB, FB, PC, BB, O) over time (000 to 007).

**Instructions:**

- add a, [\$201]
- add b, [\$202]
- add c, [\$203]
- add d, [\$204]
- add e, [\$205]
- add f, [\$206]
- add g, [\$207]
- add h, [\$208]
- add i, [\$209]
- add j, [\$20A]
- add k, [\$20B]
- add l, [\$20C]
- add m, [\$20D]
- add n, [\$20E]
- add o, [\$20F]
- add p, [\$210]
- add q, [\$211]
- add r, [\$212]
- add s, [\$213]
- add t, [\$214]
- add u, [\$215]
- add v, [\$216]
- add w, [\$217]
- add x, [\$218]
- add y, [\$219]
- add z, [\$21A]
- add aa, [\$21B]
- add ab, [\$21C]
- add ac, [\$21D]
- add ad, [\$21E]
- add ae, [\$21F]
- add af, [\$220]
- add ag, [\$221]
- add ah, [\$222]
- add ai, [\$223]
- add aj, [\$224]
- add ak, [\$225]
- add al, [\$226]
- add am, [\$227]
- add an, [\$228]
- add ao, [\$229]
- add ap, [\$22A]
- add aq, [\$22B]
- add ar, [\$22C]
- add as, [\$22D]
- add at, [\$22E]
- add au, [\$22F]
- add av, [\$230]
- add aw, [\$231]
- add ax, [\$232]
- add ay, [\$233]
- add az, [\$234]
- add ba, [\$235]
- add bb, [\$236]
- add bc, [\$237]
- add bd, [\$238]
- add be, [\$239]
- add bf, [\$23A]
- add bg, [\$23B]
- add bh, [\$23C]
- add bi, [\$23D]
- add bj, [\$23E]
- add bk, [\$23F]
- add bl, [\$240]
- add bm, [\$241]
- add bn, [\$242]
- add bo, [\$243]
- add bp, [\$244]
- add bq, [\$245]
- add br, [\$246]
- add bs, [\$247]
- add bt, [\$248]
- add bu, [\$249]
- add bv, [\$24A]
- add bw, [\$24B]
- add bx, [\$24C]
- add by, [\$24D]
- add bz, [\$24E]
- add ca, [\$24F]
- add cb, [\$250]
- add cc, [\$251]
- add cd, [\$252]
- add ce, [\$253]
- add cf, [\$254]
- add cg, [\$255]
- add ch, [\$256]
- add ci, [\$257]
- add cj, [\$258]
- add ck, [\$259]
- add cl, [\$25A]
- add cm, [\$25B]
- add cn, [\$25C]
- add co, [\$25D]
- add cp, [\$25E]
- add cq, [\$25F]
- add cr, [\$260]
- add cs, [\$261]
- add ct, [\$262]
- add cu, [\$263]
- add cv, [\$264]
- add cw, [\$265]
- add cx, [\$266]
- add cy, [\$267]
- add cz, [\$268]
- add da, [\$269]
- add db, [\$26A]
- add dc, [\$26B]
- add dd, [\$26C]
- add de, [\$26D]
- add df, [\$26E]
- add dg, [\$26F]
- add dh, [\$270]
- add di, [\$271]
- add dj, [\$272]
- add dk, [\$273]
- add dl, [\$274]
- add dm, [\$275]
- add dn, [\$276]
- add do, [\$277]
- add dp, [\$278]
- add dq, [\$279]
- add dr, [\$27A]
- add ds, [\$27B]
- add dt, [\$27C]
- add du, [\$27D]
- add dv, [\$27E]
- add dw, [\$27F]
- add dx, [\$280]
- add dy, [\$281]
- add dz, [\$282]
- add ea, [\$283]
- add eb, [\$284]
- add ec, [\$285]
- add ed, [\$286]
- add ee, [\$287]
- add ef, [\$288]
- add eg, [\$289]
- add eh, [\$28A]
- add ei, [\$28B]
- add ej, [\$28C]
- add ek, [\$28D]
- add el, [\$28E]
- add em, [\$28F]
- add en, [\$290]
- add eo, [\$291]
- add ep, [\$292]
- add eq, [\$293]
- add er, [\$294]
- add es, [\$295]
- add et, [\$296]
- add eu, [\$297]
- add ev, [\$298]
- add ew, [\$299]
- add ex, [\$29A]
- add ey, [\$29B]
- add ez, [\$29C]
- add fa, [\$29D]
- add fb, [\$29E]
- add fc, [\$29F]
- add fd, [\$2A0]
- add fe, [\$2A1]
- add ff, [\$2A2]
- add fg, [\$2A3]
- add fh, [\$2A4]
- add fi, [\$2A5]
- add fj, [\$2A6]
- add fk, [\$2A7]
- add fl, [\$2A8]
- add fm, [\$2A9]
- add fn, [\$2AA]
- add fo, [\$2AB]
- add fp, [\$2AC]
- add fq, [\$2AD]
- add fr, [\$2AE]
- add fs, [\$2AF]
- add ft, [\$2B0]
- add fu, [\$2B1]
- add fv, [\$2B2]
- add fw, [\$2B3]
- add fx, [\$2B4]
- add fy, [\$2B5]
- add fz, [\$2B6]
- add ga, [\$2B7]
- add gb, [\$2B8]
- add gc, [\$2B9]
- add gd, [\$2BA]
- add ge, [\$2BB]
- add gf, [\$2BC]
- add gg, [\$2BD]
- add gh, [\$2BE]
- add gi, [\$2BF]
- add gj, [\$2C0]
- add gk, [\$2C1]
- add gl, [\$2C2]
- add gm, [\$2C3]
- add gn, [\$2C4]
- add go, [\$2C5]
- add gp, [\$2C6]
- add gq, [\$2C7]
- add gr, [\$2C8]
- add gs, [\$2C9]
- add gt, [\$2CA]
- add gu, [\$2CB]
- add gv, [\$2CC]
- add gw, [\$2CD]
- add gx, [\$2CE]
- add gy, [\$2CF]
- add gz, [\$2D0]
- add ha, [\$2D1]
- add hb, [\$2D2]
- add hc, [\$2D3]
- add hd, [\$2D4]
- add he, [\$2D5]
- add hf, [\$2D6]
- add hg, [\$2D7]
- add hh, [\$2D8]
- add hi, [\$2D9]
- add hj, [\$2DA]
- add hk, [\$2DB]
- add hl, [\$2DC]
- add hm, [\$2DD]
- add hn, [\$2DE]
- add ho, [\$2DF]
- add hp, [\$2E0]
- add hq, [\$2E1]
- add hr, [\$2E2]
- add hs, [\$2E3]
- add ht, [\$2E4]
- add hu, [\$2E5]
- add hv, [\$2E6]
- add hw, [\$2E7]
- add hx, [\$2E8]
- add hy, [\$2E9]
- add hz, [\$2EA]
- add ia, [\$2EB]
- add ib, [\$2EC]
- add ic, [\$2ED]
- add id, [\$2EE]
- add ie, [\$2EF]
- add if, [\$2F0]
- add ig, [\$2F1]
- add ih, [\$2F2]
- add ii, [\$2F3]
- add ij, [\$2F4]
- add ik, [\$2F5]
- add il, [\$2F6]
- add im, [\$2F7]
- add in, [\$2F8]
- add io, [\$2F9]
- add ip, [\$2FA]
- add iq, [\$2FB]
- add ir, [\$2FC]
- add is, [\$2FD]
- add it, [\$2FE]
- add iu, [\$2FF]
- add iv, [\$300]
- add iw, [\$301]
- add ix, [\$302]
- add iy, [\$303]
- add iz, [\$304]
- add ja, [\$305]
- add jb, [\$306]
- add jc, [\$307]
- add jd, [\$308]
- add je, [\$309]
- add jf, [\$30A]
- add jg, [\$30B]
- add jh, [\$30C]
- add ji, [\$30D]
- add jj, [\$30E]
- add jk, [\$30F]
- add jl, [\$310]
- add jm, [\$311]
- add jn, [\$312]
- add jo, [\$313]
- add jp, [\$314]
- add jq, [\$315]
- add jr, [\$316]
- add js, [\$317]
- add jt, [\$318]
- add ju, [\$319]
- add jv, [\$31A]
- add jw, [\$31B]
- add jx, [\$31C]
- add jy, [\$31D]
- add jz, [\$31E]
- add ka, [\$31F]
- add kb, [\$320]
- add kc, [\$321]
- add kd, [\$322]
- add ke, [\$323]
- add kf, [\$324]
- add kg, [\$325]
- add kh, [\$326]
- add ki, [\$327]
- add kj, [\$328]
- add kk, [\$329]
- add kl, [\$32A]
- add km, [\$32B]
- add kn, [\$32C]
- add ko, [\$32D]
- add kp, [\$32E]
- add kq, [\$32F]
- add kr, [\$330]
- add ks, [\$331]
- add kt, [\$332]
- add ku, [\$333]
- add kv, [\$334]
- add kw, [\$335]
- add kx, [\$336]
- add ky, [\$337]
- add kz, [\$338]
- add la, [\$339]
- add lb, [\$33A]
- add lc, [\$33B]
- add ld, [\$33C]
- add le, [\$33D]
- add lf, [\$33E]
- add lg, [\$33F]
- add lh, [\$340]
- add li, [\$341]
- add lj, [\$342]
- add lk, [\$343]
- add ll, [\$344]
- add lm, [\$345]
- add ln, [\$346]
- add lo, [\$347]
- add lp, [\$348]
- add lq, [\$349]
- add lr, [\$34A]
- add ls, [\$34B]
- add lt, [\$34C]
- add lu, [\$34

**Overflow Handling**

**Store Skips On Overflow**

Assembly code snippets:

```

ld a, [$200]
add a, $201
ld [$202], a
jmp cont
ld [$203], a

```

Register Table:

A	0001
B	0F00
LR	05AA
EB	0000
FB	0000
PC	0421
BB	0000
0	0000

Diagram illustrating an addition operation:

```

  0000
+ 0000
-----
  0000

```

Memory dump (hex values):

7FFF	200
0001	201
0000	202
0001	203

Double Word Result

Count, Compare and Skip

ccs [k]	CCS k
1000+k	2
<pre>A ← DABS(M[k]) if M[k]&gt;+0:     PC ← PC if M[k]=+0:     PC ← PC+1 if M[k]&lt;-0:     PC ← PC+2 if M[k]=-0:     PC ← PC+3</pre>	

CCS

1200	41F	ccs [\$200]	>+0
0430	420	jmp \$430	=+0
0440	421	jmp \$440	<-0
0450	422	jmp \$450	=-0
3080	62E	ld a, [\$80]	

Editing

ROR 1010111100100101

# Shift/Rotate Using Memory

SHR8 0101111001001011

**I/O Channels**

**B**  
**LR**

000 00  
001 01  
002 02  
003 03  
004 04  
005 05  
006 06  
007 07  
008 08  
009 09  
00A 0A  
00B 0B  
00C 0C  
00D 0D  
00E 0E  
00F 0F

# Boolean Using I/O Instructions

In	in a, [kc]	READ kc
	0006, 0000+kc	2
	A ← IO[kc]	

Out	out [kc], a	WRITE kc
	0006, 0200+kc	2
	IO[kc] ← A	

In	in& a, [kc]	RAND kc
	0006, 0000+kc	2
	A ← A & B	

Out	out& [kc], a	WAND kc
	0006, 0200+kc	2
	A, IO[kc] ← A & IO[kc]	

In	in  a, [kc]	ROR kc
	0006, 0000+kc	2
	A ← A   B	

Out	out  [kc], a	WOR kc
	0006, 0200+kc	2
	A, IO[kc] ← A   IO[kc]	

In XOR	in^ a, [kc]	RXOR kc
	0006, 0C00+kc	2
	A ← A ^ B	

**INDEX**

**Interrupt**

**Indexing**

INDEX k

INDEX k

6000+k

2

$IR \leftarrow IR + M[k]$

PC →

6080

3700

**INDEX [\$80]**

**ld a, [\$702]**

IR 3700

+

0002

IR 3702

0CDC

3193

4584

0002

0400

37B5

52A3

24D1

Diagram illustrating the indexing process for the instruction `ld a, [$700+[$80]]`.

The diagram shows three memory tables:

- Base Table (Address Range \$700 - \$706):** Contains values 0000, 0000, 0000, 0003, 0006, 0009, 000C, 000F, 0012.
- Index Table (Address Range \$80 - \$84):** Contains values 0700, 0002, 0702, 23C9, 0002, 291C, 20F7, 100F, 45B3.
- Final Result Table (Address Range \$7C - \$84):** Contains values 07C, 33DF, 1093, 23C9, 0002, 291C, 20F7, 100F, 45B3.

The process involves adding the base value (0006) and the index value (0700) to get the final address (0706).

The final result table shows the values stored at the final addresses, including the value 0002 at address \$80.



## Quirks

## One's Complement

## Overflow Disables Interrupts

## Store Skips On Overflow

## CCS

## Shift/Rotate Using Memory

## Boolean Using I/O Instructions

## Indexing By IR Addition

## Indexing By IR Addition

## Indexing



Quirks

One's Complement

One's Complement

Interrupts & Overflow

Overflow Disables Interrupts

A	0000
B	0F00
LR	05AA
EB	0000
FB	0000
PC	0421
BB	0000
0	0000

add a, [\$201]  
nop  
ld [\$202], a  
ld [\$203], a

Overflow Handling

Store Skips On Overflow

A	0001
B	0F00
LR	05AA
EB	0000
FB	0000
PC	0421
BB	0000
0	0000

ld a, [\$200]  
ld [\$202], a  
jmp cont  
ld [\$203], a

Conditional Jumps

CCS

ccs [k]	CCS k
1000+k	2

A ← DABS(M[k])  
if M[k]>+0:  
PC ← PC  
if M[k]=+0:  
PC ← PC+1  
if M[k]<-0:  
PC ← PC+2  
if M[k]=-0:  
PC ← PC+3

jmp \$430  
jmp \$440  
jmp \$450  
ld a, [\$80]

Editing

Shift/Rotate Using Memory

ROR 101011100100101

SHR8 010111001001011

I/O Channels

Boolean Using I/O Instructions

In	Out
in a, [kc] READ kc	out [kc], a WRITE kc
0006, 0000+kc 2	0006, 0200+kc 2
A ← IO[kc]	IO[kc] ← A
in& a, [kc] RAND kc	out& [kc], a WAND kc
0006, 0000+kc 2	0006, 0200+kc 2
A ← A & IO[kc]	A, IO[kc] ← A & IO[kc]
in  a, [kc] ROR kc	out  [kc], a WOR kc
0006, 0000+kc 2	0006, 0200+kc 2
A ← A   IO[kc]	A, IO[kc] ← A   IO[kc]
in^ a, [kc] RXOR kc	
0006, 0C00+kc 2	
A ← A ^ IO[kc]	

INDEX

Indexing

Interrupt

INDEX k	INDEX k
6000+k	2

IR ← IR + M[k]

IR 3702

Indexing

No Stack

ld a, [\$700+\$80]

Table at \$700

A	0006
B	0F00
LR	05AA
EB	0000
FB	0000
PC	0421
BB	0000
0	0000

0000	0700	133A
0000	0002	33DF
0000	0702	1093
0003		3C9
0009		291C
000C		20F7
000F		100F
0012		45B3

**Architecture**

**Instruction Set**

**Memory Model**

**Arithmetic**

**I/O**

**Interrupts**

**Encoding**

**Counters**

**Architecture**

# Apollo Guidance Computer

Architecture

Mission Software

Hardware

Peripherals

System Software

**Hardware**



Hardware

**Hardware**

**1 MHz**

**Microcode**

**Integrated  
Circuits**

**Core Memory**

**Core Rope  
Memory**

**Hardware**

**Hardware**

**Block Diagram**

**Microcode**

**Hardware**

**Block Diagram**

**Schematics**

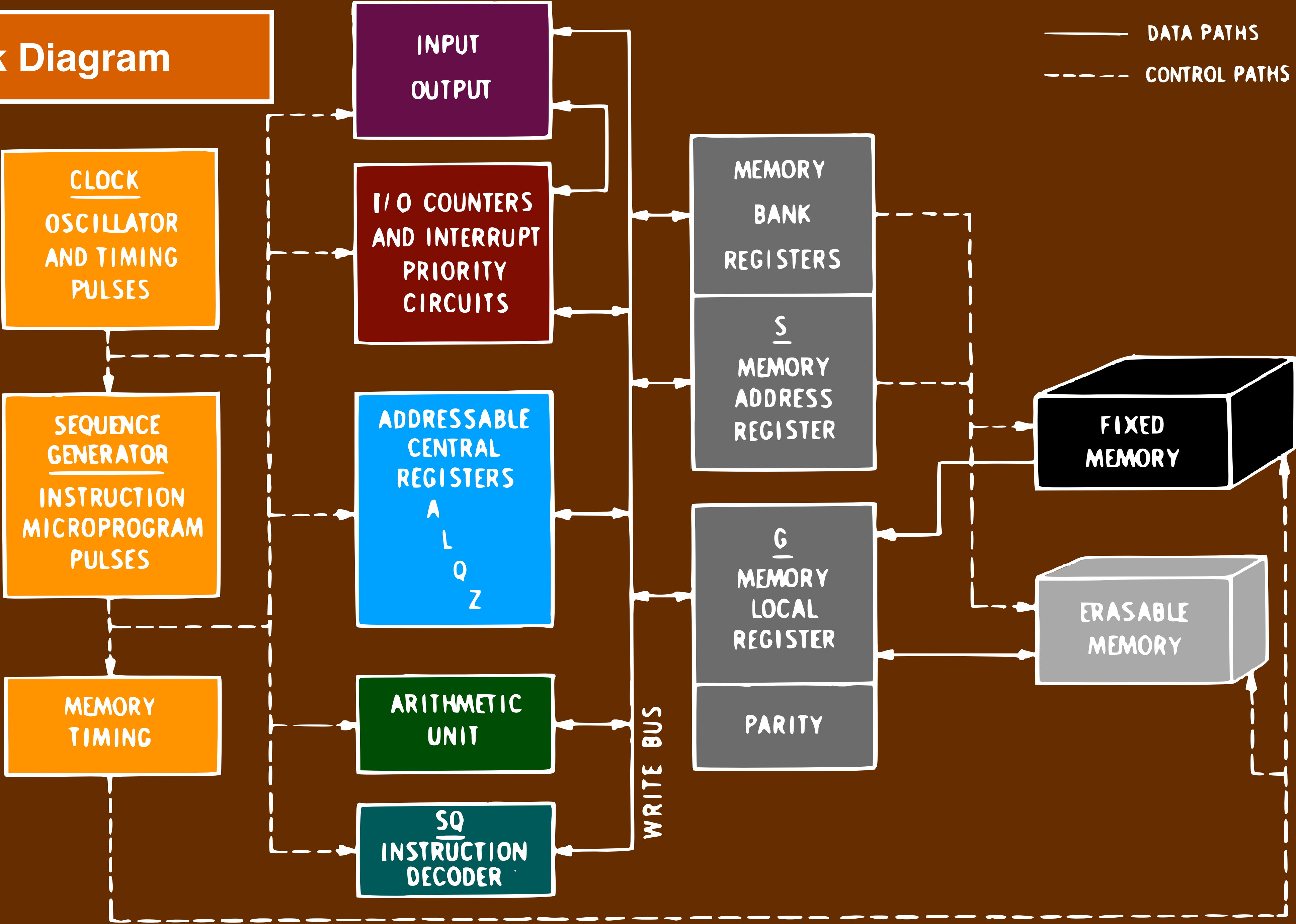
**Microcode**

**ICs**

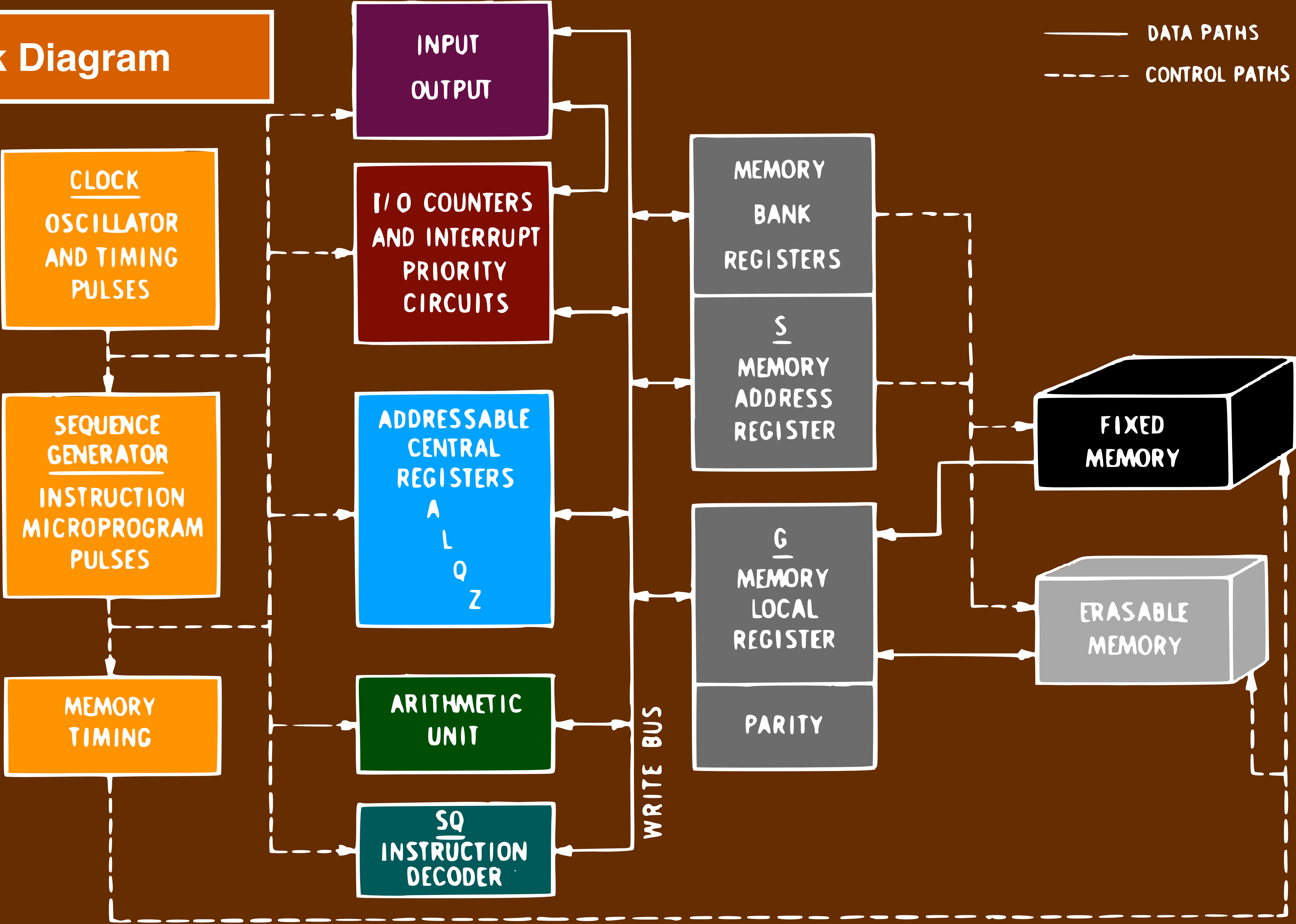
**Memory**



# Block Diagram



# Block Diagram



# Block Diagram

CLOCK  
OSCILLATOR  
AND TIMING  
PULSES

SEQUENCE  
GENERATOR  
INSTRUCTION  
MICROPROGRAM  
PULSES

MEMORY  
TIMING

INPUT  
OUTPUT

I/O COUNTERS  
AND INTERRUPT  
PRIORITY  
CIRCUITS

ADDRESSABLE  
CENTRAL  
REGISTERS  
A  
L  
Q  
Z

ARITHMETIC  
UNIT

SQ  
INSTRUCTION  
DECODER

MEMORY  
BANK  
REGISTERS

S  
MEMORY  
ADDRESS  
REGISTER

G  
MEMORY  
LOCAL  
REGISTER  
PARITY

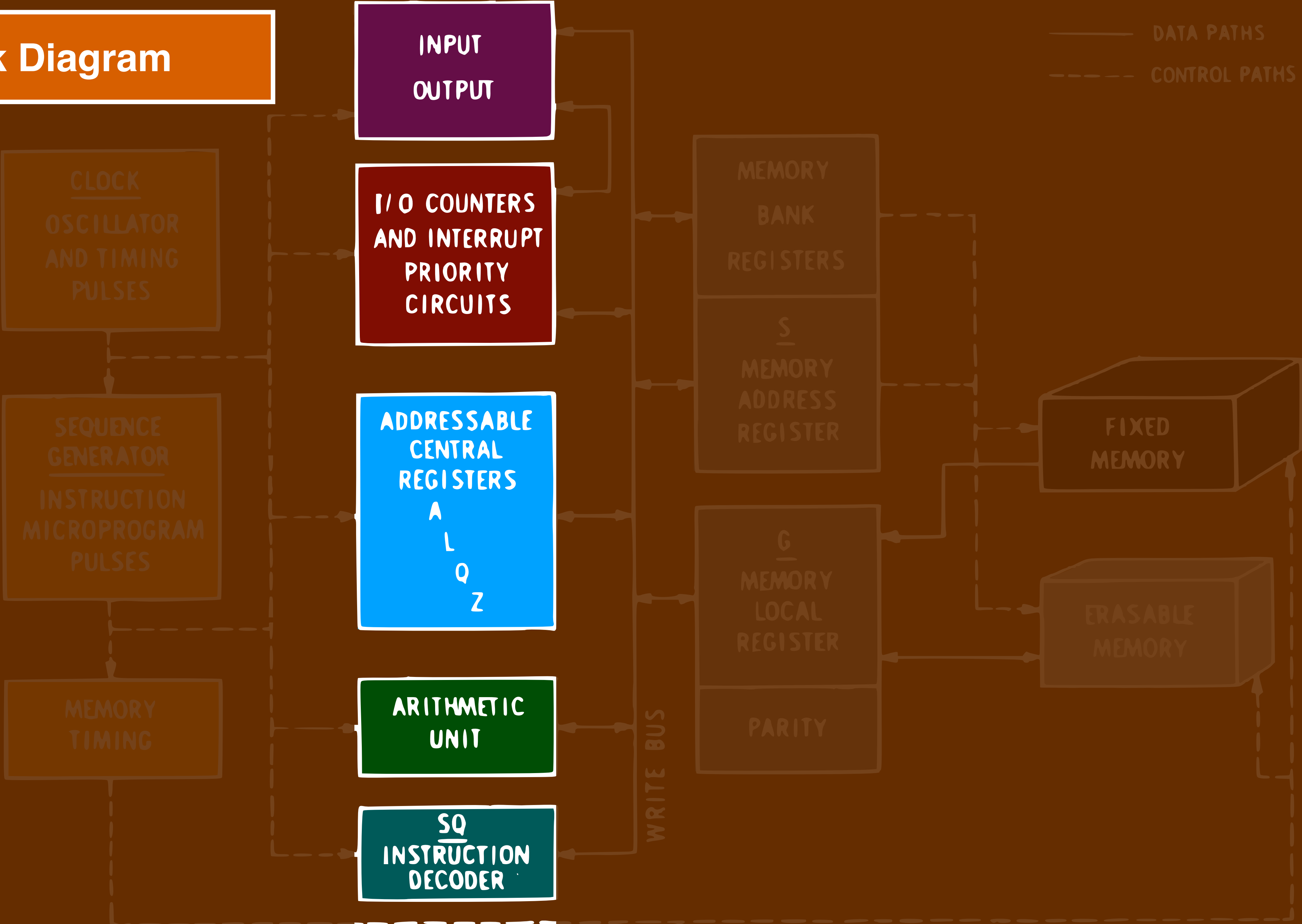
DATA PATHS  
CONTROL PATHS

FIXED  
MEMORY

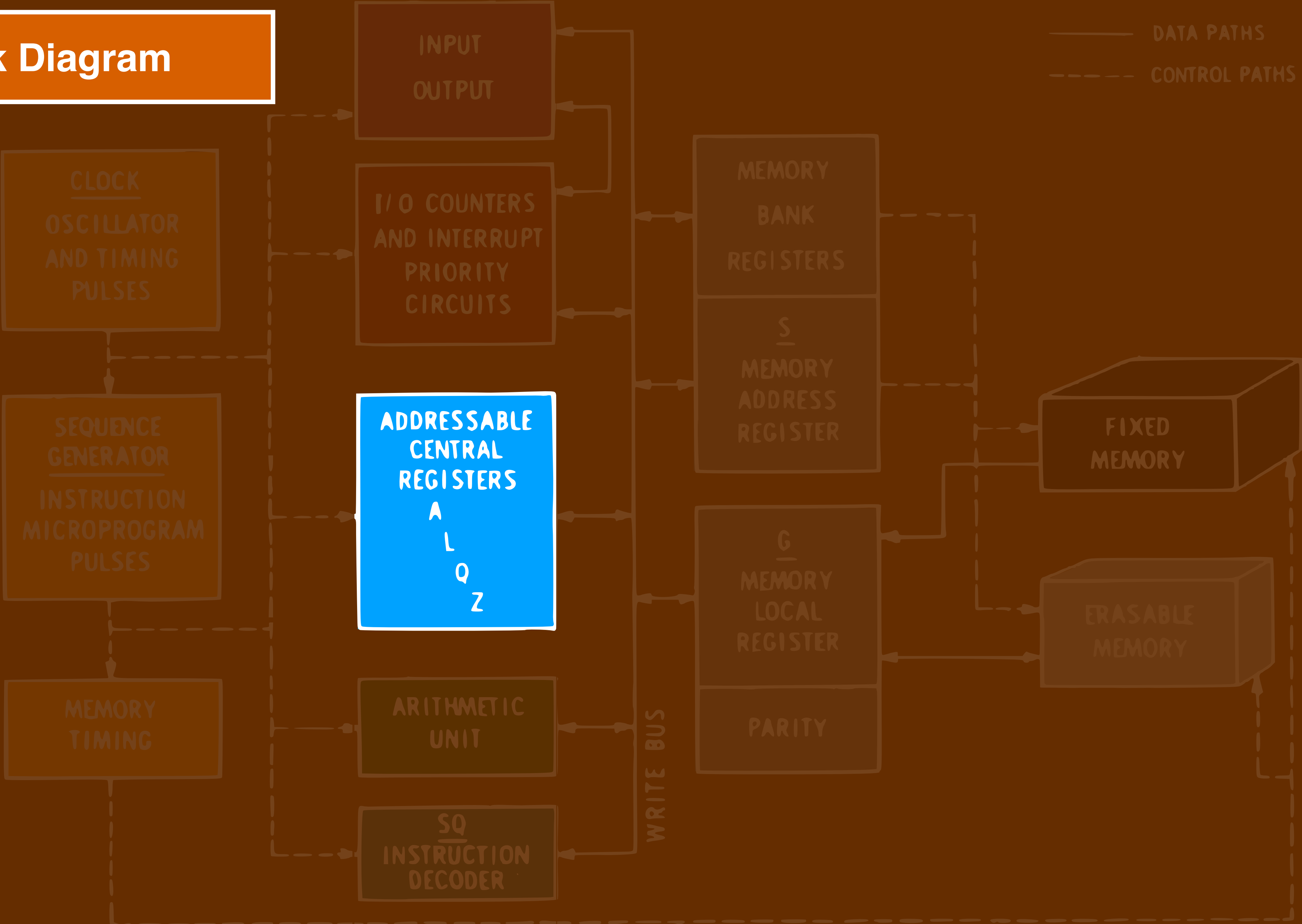
ERASABLE  
MEMORY

WRITE BUS

# Block Diagram

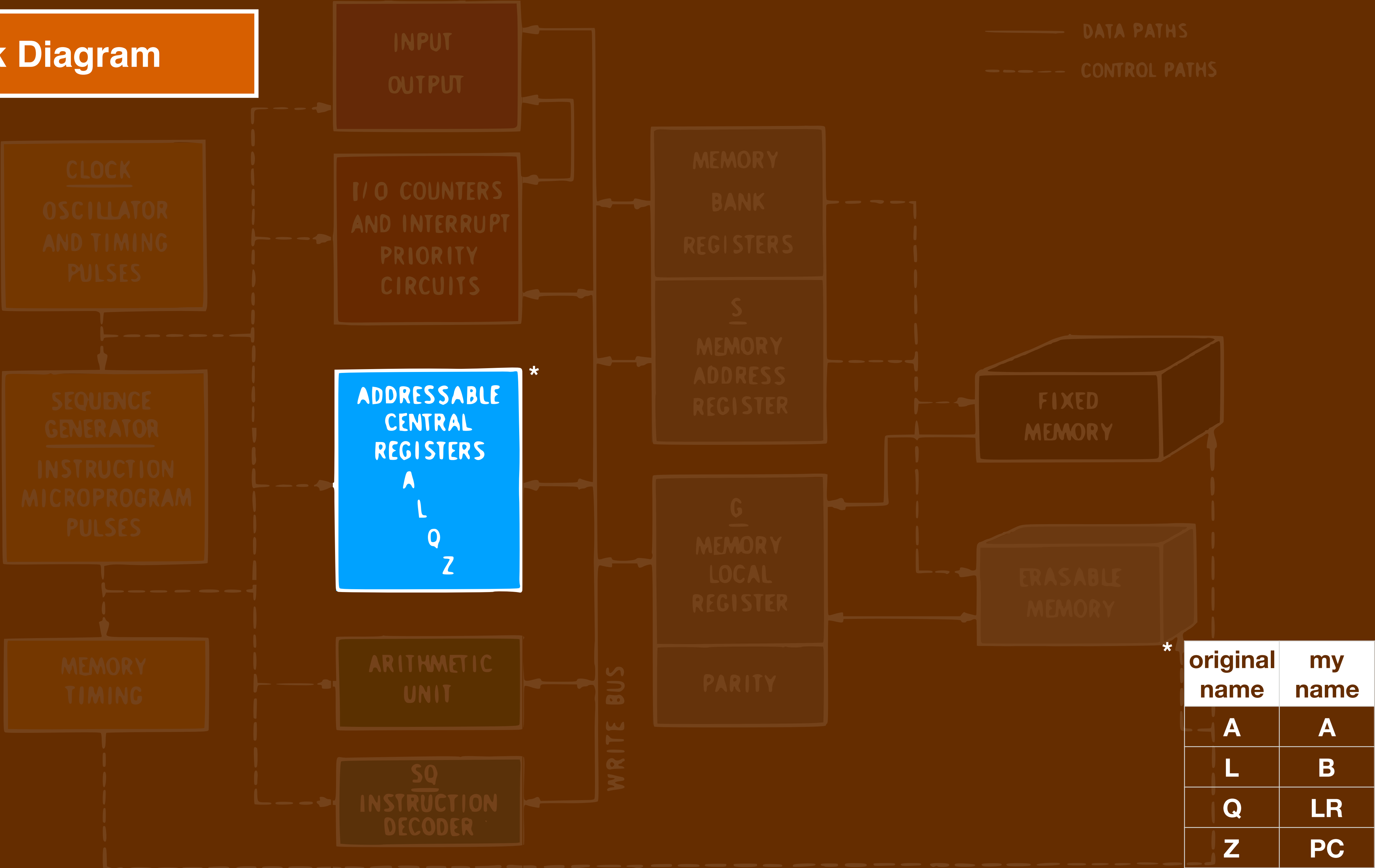


# Block Diagram



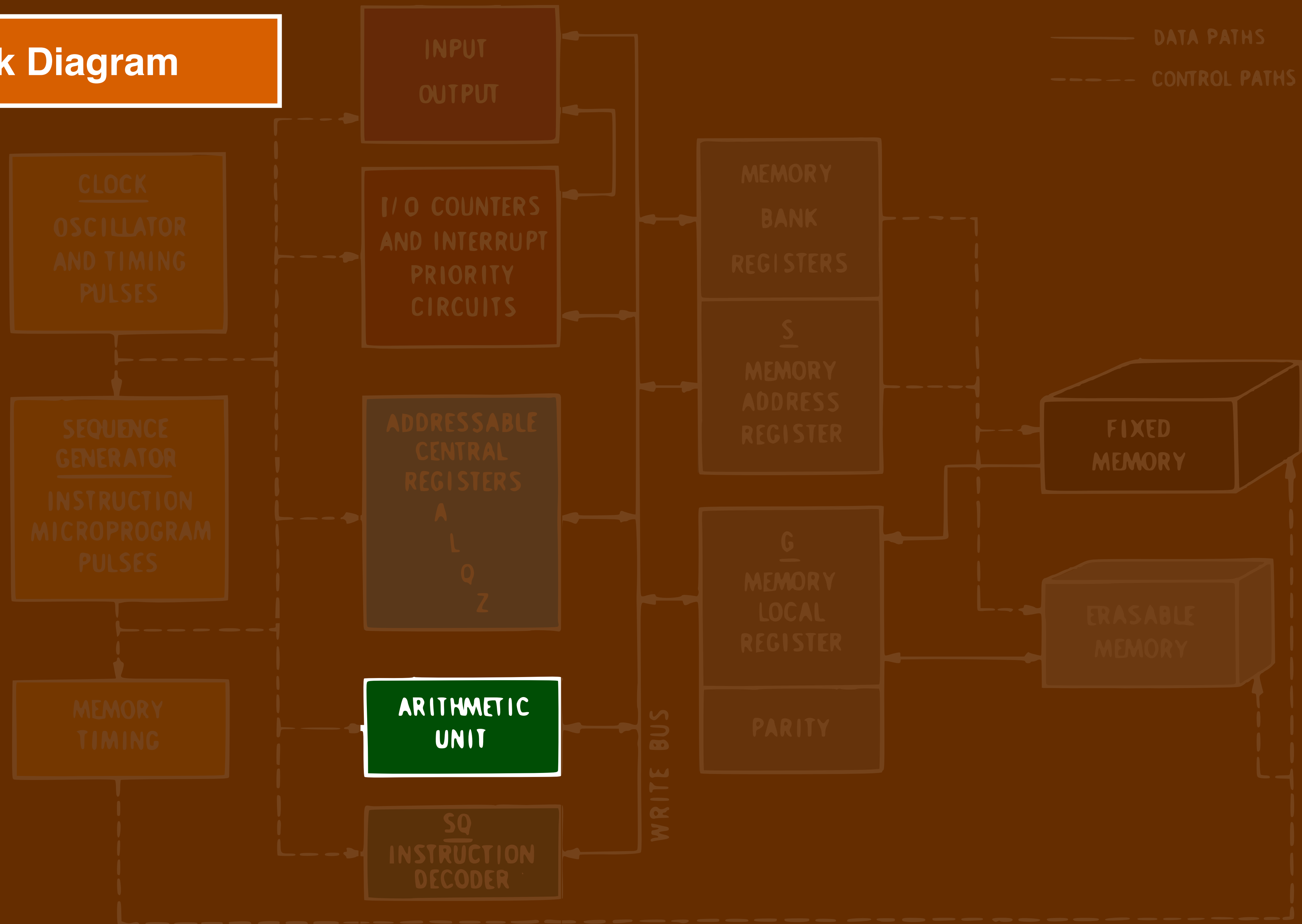


# Block Diagram

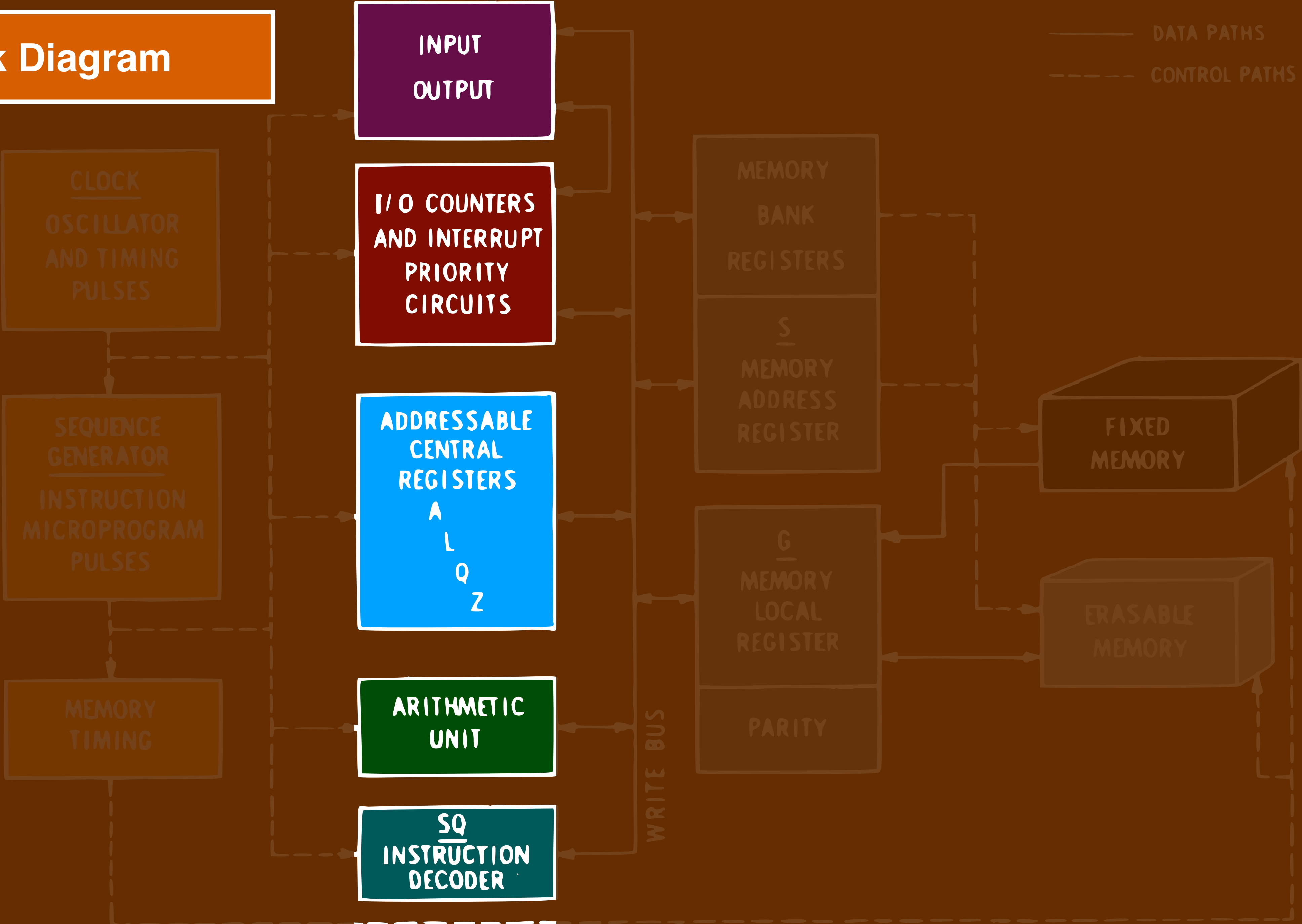


original name	my name
A	A
L	B
Q	LR
Z	PC

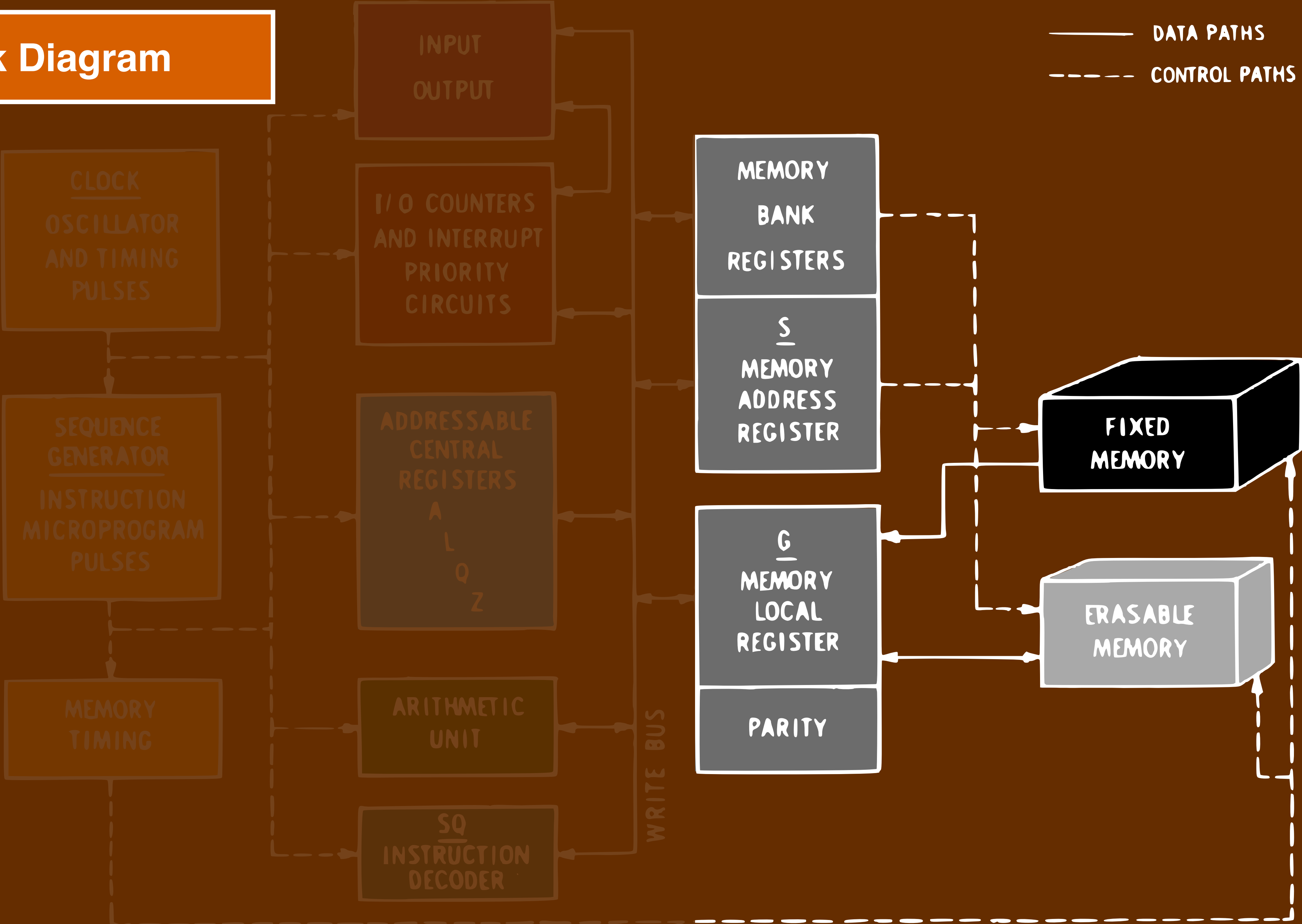
# Block Diagram



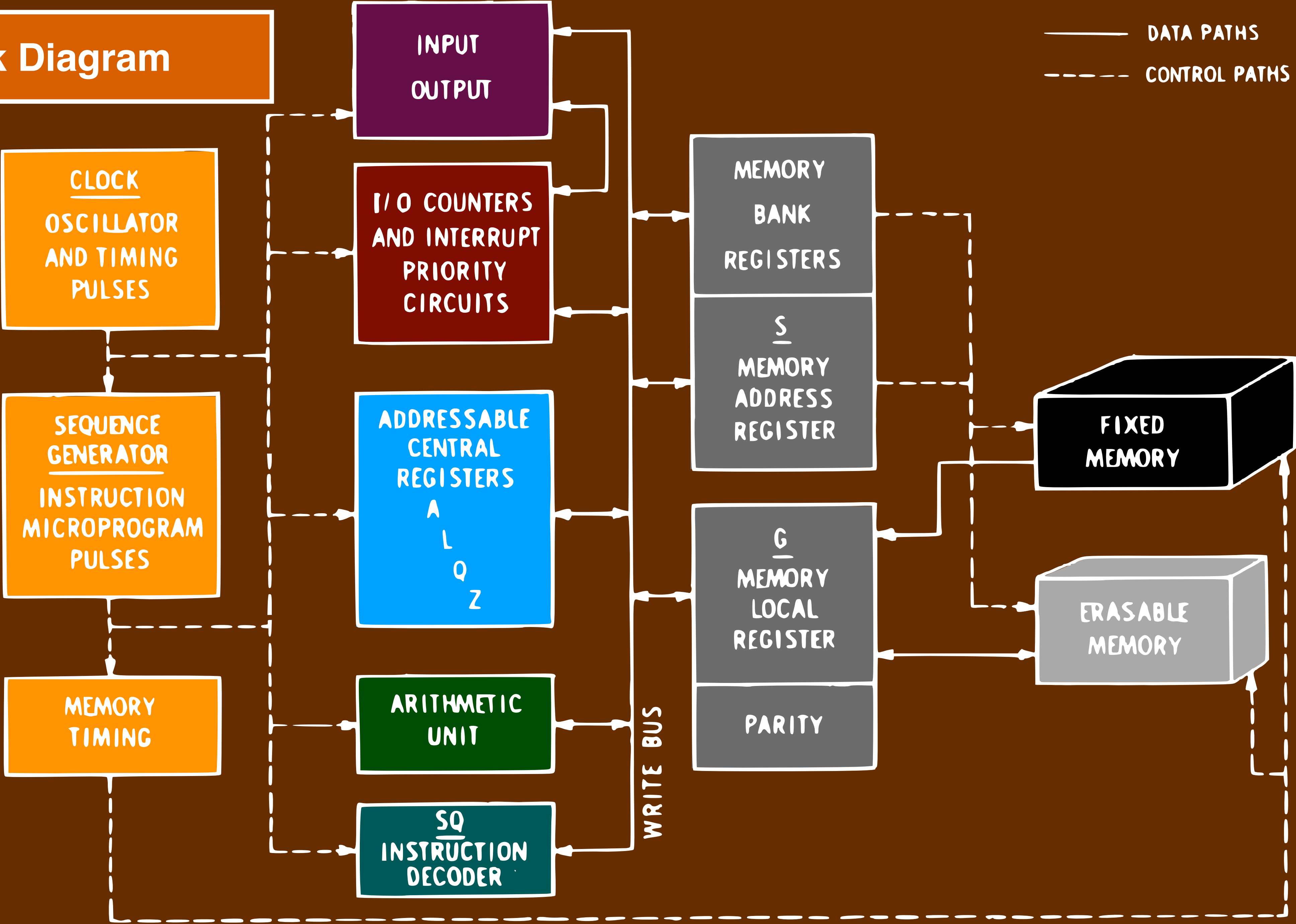
# Block Diagram



# Block Diagram



# Block Diagram





Clock

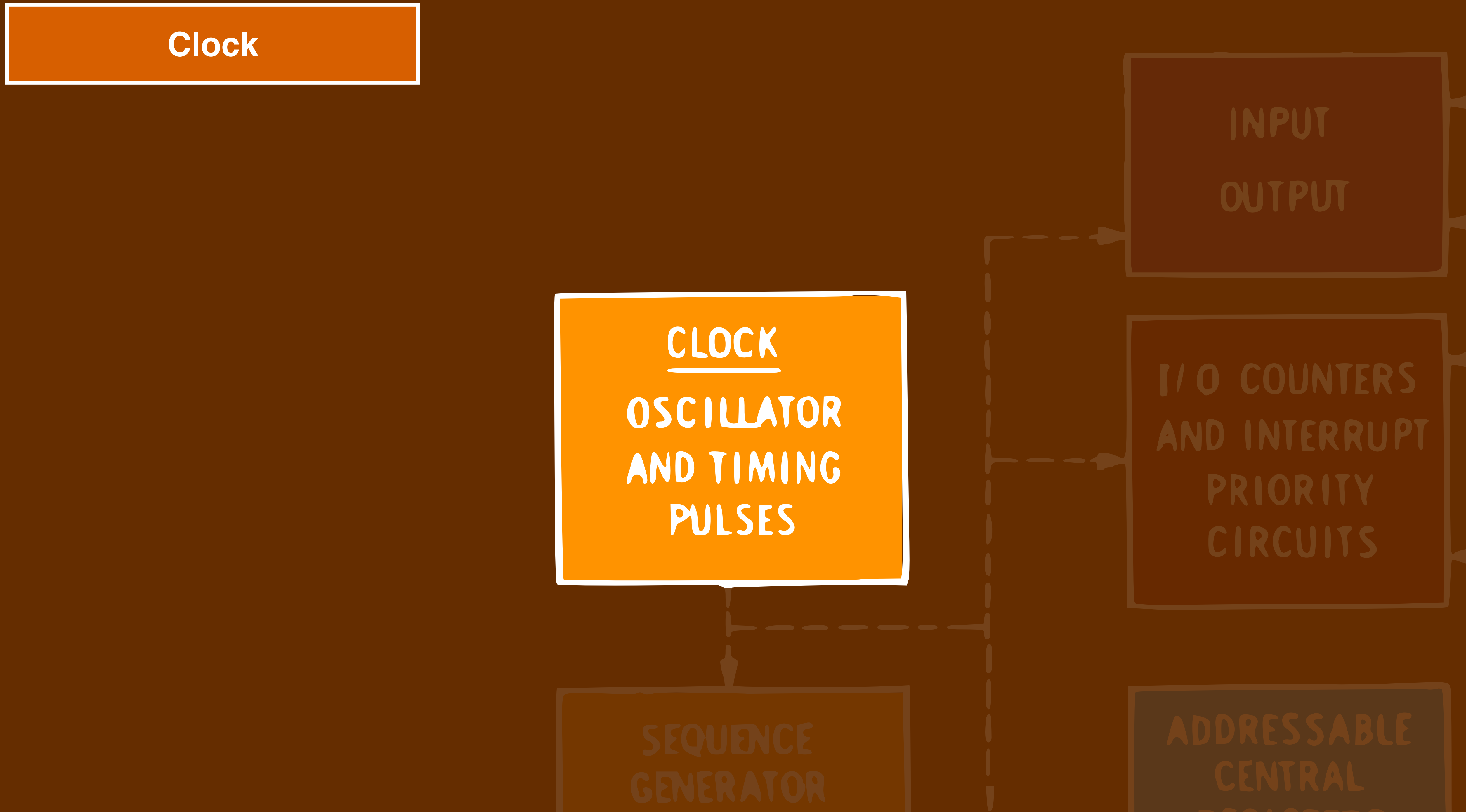
CLOCK  
OSCILLATOR  
AND TIMING  
PULSES

INPUT  
OUTPUT

I/O COUNTERS  
AND INTERRUPT  
PRIORITY  
CIRCUITS

SEQUENCE  
GENERATOR

ADDRESSABLE  
CENTRAL  
PROCESSOR



Clock

1.024 MHz

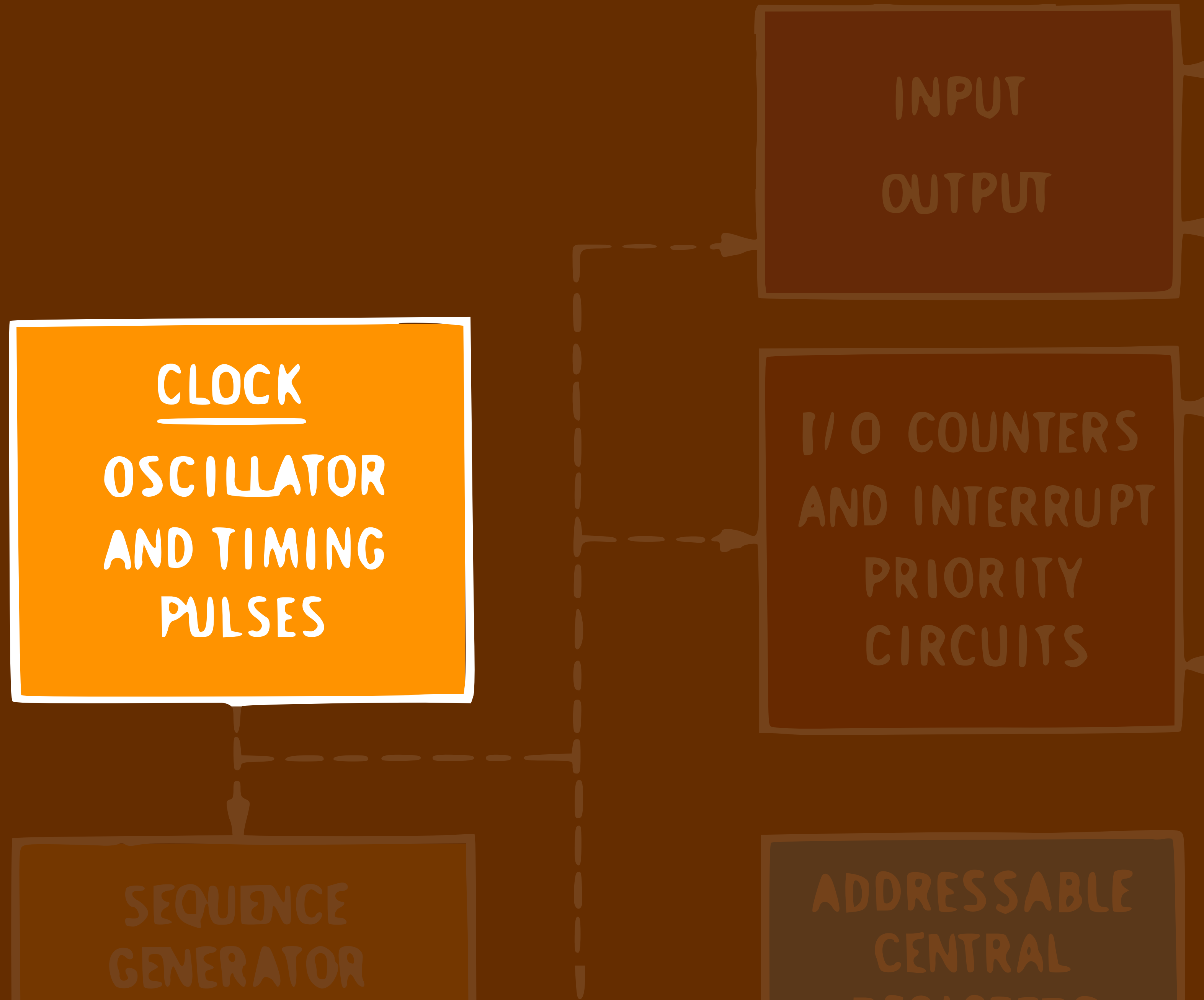
CLOCK  
OSCILLATOR  
AND TIMING  
PULSES

INPUT  
OUTPUT

I/O COUNTERS  
AND INTERRUPT  
PRIORITY  
CIRCUITS

SEQUENCE  
GENERATOR

ADDRESSABLE  
CENTRAL  
PROCESSOR



Sequencer

OSCILLATOR  
AND TIMING  
PULSES

AND INTERRUPT  
PRIORITY  
CIRCUITS

SEQUENCE  
GENERATOR  
INSTRUCTION  
MICROPROGRAM  
PULSES

ADDRESSABLE  
CENTRAL  
REGISTERS

A  
L  
Q  
Z

MEMORY  
TIMING

ARITHMETIC  
UNIT

Sequencer

- T9
- T10
- T11
- T12
- T1
- T2
- T3
- T4
- T5
- T6
- T7
- T8
- T9
- T10
- T11
- T12
- T1
- T2
- T3
- T4
- T5

1 MCT

SEQUENCE  
GENERATOR  
INSTRUCTION  
MICROPROGRAM  
PULSES

OSCILLATOR  
AND TIMING  
PULSES

AND INTERRUPT  
PRIORITY  
CIRCUITS

ADDRESSABLE  
CENTRAL  
REGISTERS

A  
L  
Q  
Z

MEMORY  
TIMING

ARITHMETIC  
UNIT

Sequencer

- T9
- T10
- T11
- T12
- T1
- T2
- T3
- T4
- T5
- T6
- T7
- T8
- T9
- T10
- T11
- T12
- T1
- T2
- T3
- T4
- T5

1 MCT

SEQUENCE  
GENERATOR  
INSTRUCTION  
MICROPROGRAM  
PULSES

MCT

1	ld a, [\$200]
2	
3	add a, [\$201]
4	
5	ld [\$202], a
6	
7	jmp cont
8	ld a, [\$203]
9	



Sequencer

ld a, [k]

T9				
T10				
T11				
T12				
T1				
T2	RSC	WG		
T3				
T4				
T5				
T6				
T7	RG	WB		
T8	RZ	WS	ST2	
T9	RB	WG		
T10	RB	WA		
T11				
T12				
T1				
T2				
T3				
T4				
T5				

1 MCT

SEQUENCE  
GENERATOR  
INSTRUCTION  
MICROPROGRAM  
PULSES

MCT

1	ld a, [\$200]
2	
3	add a, [\$201]
4	
5	ld [\$202], a
6	
7	jmp cont
8	ld a, [\$203]
9	

Sequencer

ld a, [k]

T9				
T10				
T11				
T12				
T1				
T2	RSC	WG		
T3				
T4				
T5				
T6				
T7	RG	WB		
T8	RZ	WS	ST2	
T9	RB	WG		
T10	RB	WA		
T11				
T12				
T1				
T2				
T3				
T4				
T5				

1 MCT

SEQUENCE  
GENERATOR  
INSTRUCTION  
MICROPROGRAM  
PULSES

MCT

1	ld a, [\$200]
2	
3	add a, [\$201]
4	
5	ld [\$202], a
6	
7	jmp cont
8	ld a, [\$203]
9	

# Units

CLOCK  
OSCILLATOR  
AND TIMING  
PULSES

SEQUENCE  
GENERATOR  
INSTRUCTION  
MICROPROGRAM  
PULSES

MEMORY  
TIMING

INPUT  
OUTPUT

I/O COUNTERS  
AND INTERRUPT  
PRIORITY  
CIRCUITS

ADDRESSABLE  
CENTRAL  
REGISTERS  
A  
L  
Q  
Z

ARITHMETIC  
UNIT

SQ  
INSTRUCTION  
DECODER

MEMORY  
BANK  
REGISTERS

S  
MEMORY  
ADDRESS  
REGISTER

G  
MEMORY  
LOCAL  
REGISTER

PARITY

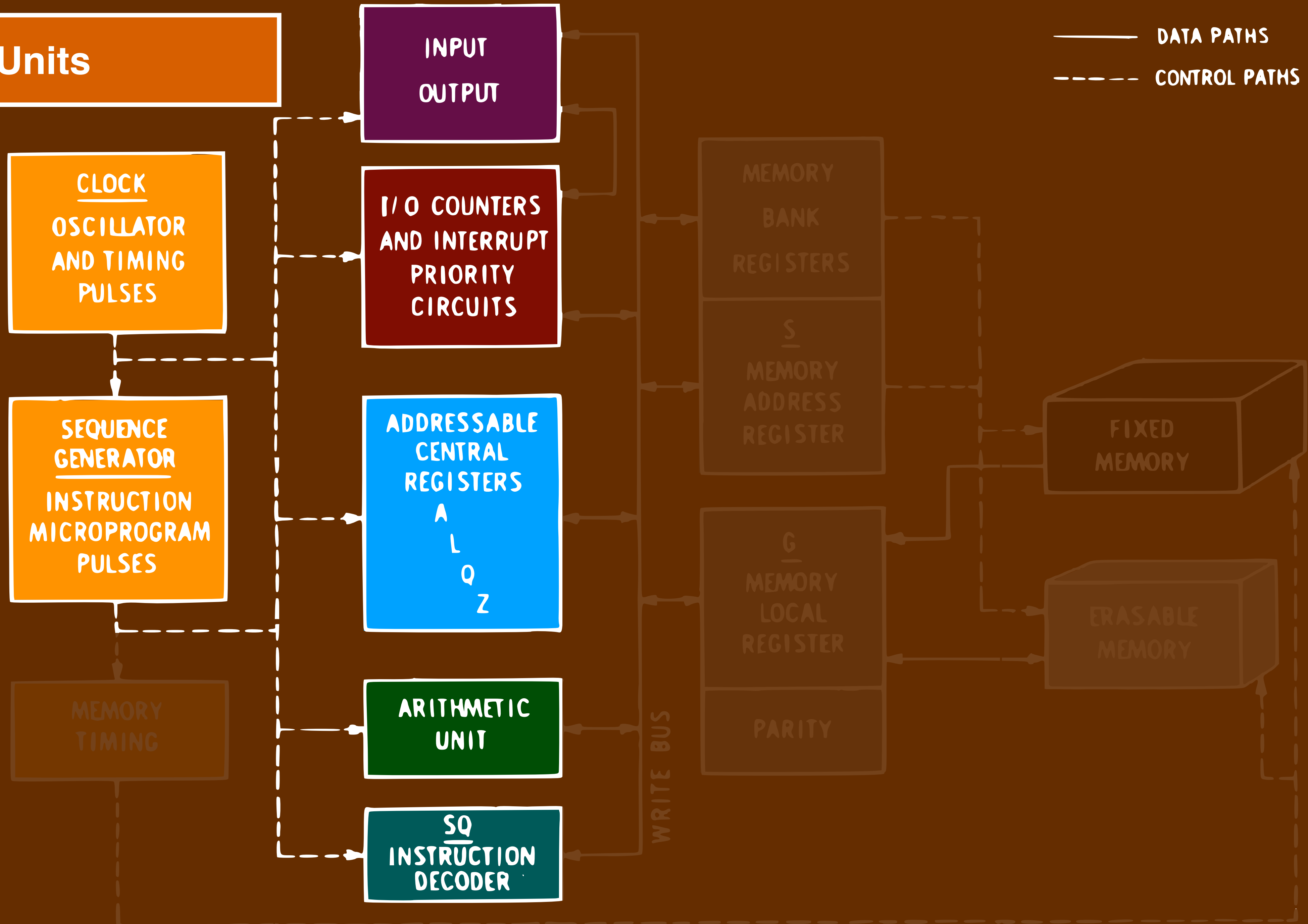
FIXED  
MEMORY

ERASABLE  
MEMORY

DATA PATHS

CONTROL PATHS

WRITE BUS



Write Bus

CLOCK  
OSCILLATOR  
AND TIMING  
PULSES

SEQUENCE  
GENERATOR  
INSTRUCTION  
MICROPROGRAM  
PULSES

MEMORY  
TIMING

INPUT  
OUTPUT

I/O COUNTERS  
AND INTERRUPT  
PRIORITY  
CIRCUITS

ADDRESSABLE  
CENTRAL  
REGISTERS  
A  
L  
Q  
Z

ARITHMETIC  
UNIT

SQ  
INSTRUCTION  
DECODER

MEMORY  
BANK  
REGISTERS

S  
MEMORY  
ADDRESS  
REGISTER

G  
MEMORY  
LOCAL  
REGISTER

PARITY

FIXED  
MEMORY

ERASABLE  
MEMORY

DATA PATHS

CONTROL PATHS

WRITE BUS

Write Bus

CLOCK  
OSCILLATOR  
AND TIMING  
PULSES

SEQUENCE  
GENERATOR  
INSTRUCTION  
MICROPROGRAM  
PULSES

MEMORY  
TIMING

INPUT  
OUTPUT

I/O COUNTERS  
AND INTERRUPT  
PRIORITY  
CIRCUITS

ADDRESSABLE  
CENTRAL  
REGISTERS  
A  
L  
Q  
Z

ARITHMETIC  
UNIT

SQ  
INSTRUCTION  
DECODER

MEMORY  
BANK  
REGISTERS

S  
MEMORY  
ADDRESS  
REGISTER

G  
MEMORY  
LOCAL  
REGISTER

PARITY

FIXED  
MEMORY

ERASABLE  
MEMORY

DATA PATHS

CONTROL PATHS

WRITE BUS



Registers

ON  
RAM

ADDRESSABLE  
CENTRAL  
REGISTERS

A

L

Q

Z

M  
A  
R  
  
M  
L  
R

Registers

ON  
RAM

RA  
----->

ADDRESSABLE  
CENTRAL  
REGISTERS

A

L

Q

Z



Registers

ON  
RAM

WA  
---

ADDRESSABLE  
CENTRAL  
REGISTERS

A

L

Q

Z



Memory

CLOCK  
OSCILLATOR  
AND TIMING  
PULSES

SEQUENCE  
GENERATOR  
INSTRUCTION  
MICROPROGRAM  
PULSES

MEMORY  
TIMING

INPUT  
OUTPUT

I/O COUNTERS  
AND INTERRUPT  
PRIORITY  
CIRCUITS

ADDRESSABLE  
CENTRAL  
REGISTERS  
A  
L  
Q  
Z

ARITHMETIC  
UNIT

SQ  
INSTRUCTION  
DECODER

MEMORY  
BANK  
REGISTERS

S  
MEMORY  
ADDRESS  
REGISTER

G  
MEMORY  
LOCAL  
REGISTER

PARITY

FIXED  
MEMORY

ERASABLE  
MEMORY

DATA PATHS

CONTROL PATHS

WRITE BUS

Memory

CLOCK  
OSCILLATOR  
AND TIMING  
PULSES

SEQUENCE  
GENERATOR  
INSTRUCTION  
MICROPROGRAM  
PULSES

MEMORY  
TIMING

INPUT  
OUTPUT

I/O COUNTERS  
AND INTERRUPT  
PRIORITY  
CIRCUITS

ADDRESSABLE  
CENTRAL  
REGISTERS  
A  
L  
Q  
Z

ARITHMETIC  
UNIT

SQ  
INSTRUCTION  
DECODER

MEMORY  
BANK  
REGISTERS

S  
MEMORY  
ADDRESS  
REGISTER

G  
MEMORY  
LOCAL  
REGISTER

PARITY

FIXED  
MEMORY

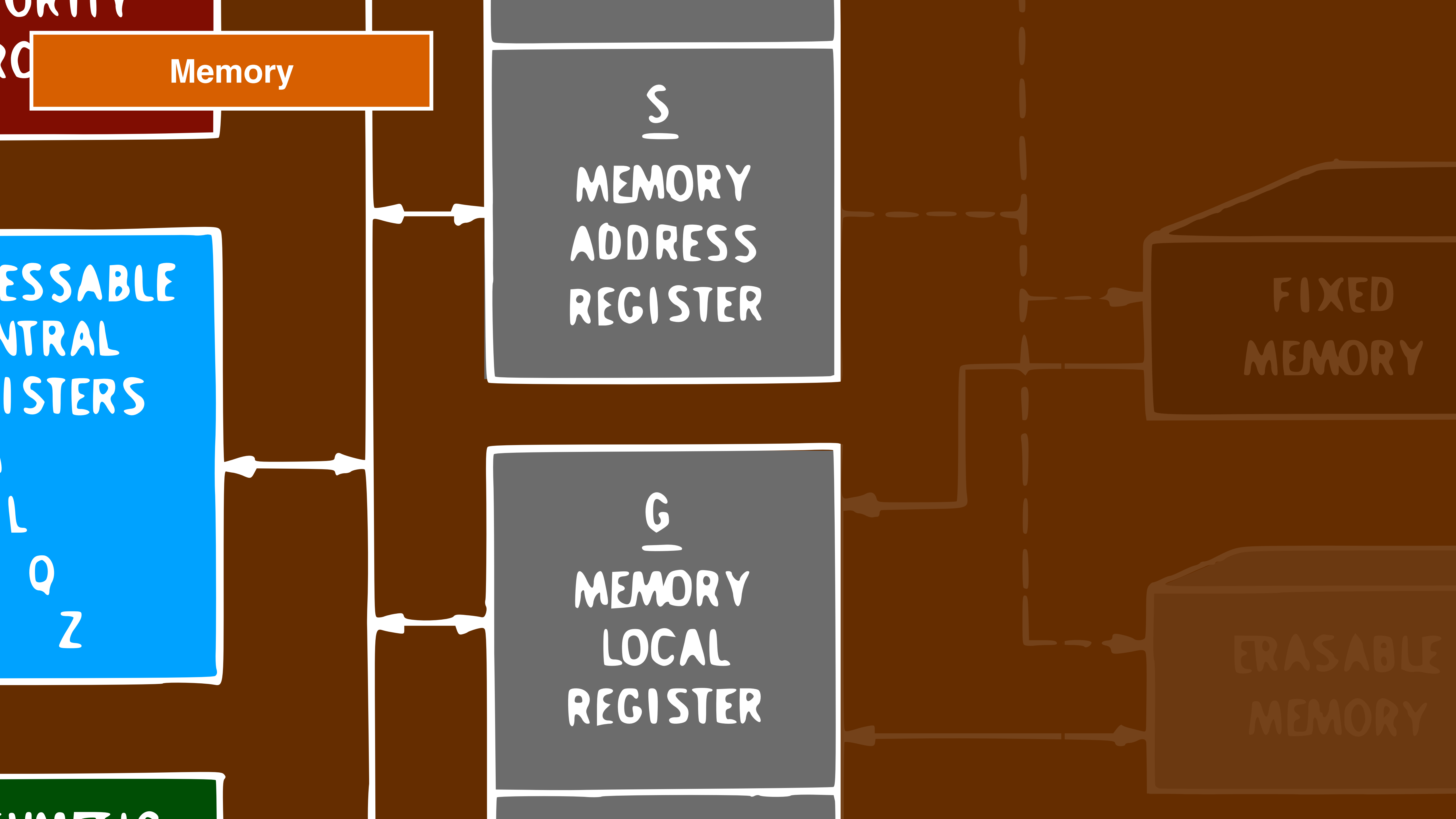
ERASABLE  
MEMORY

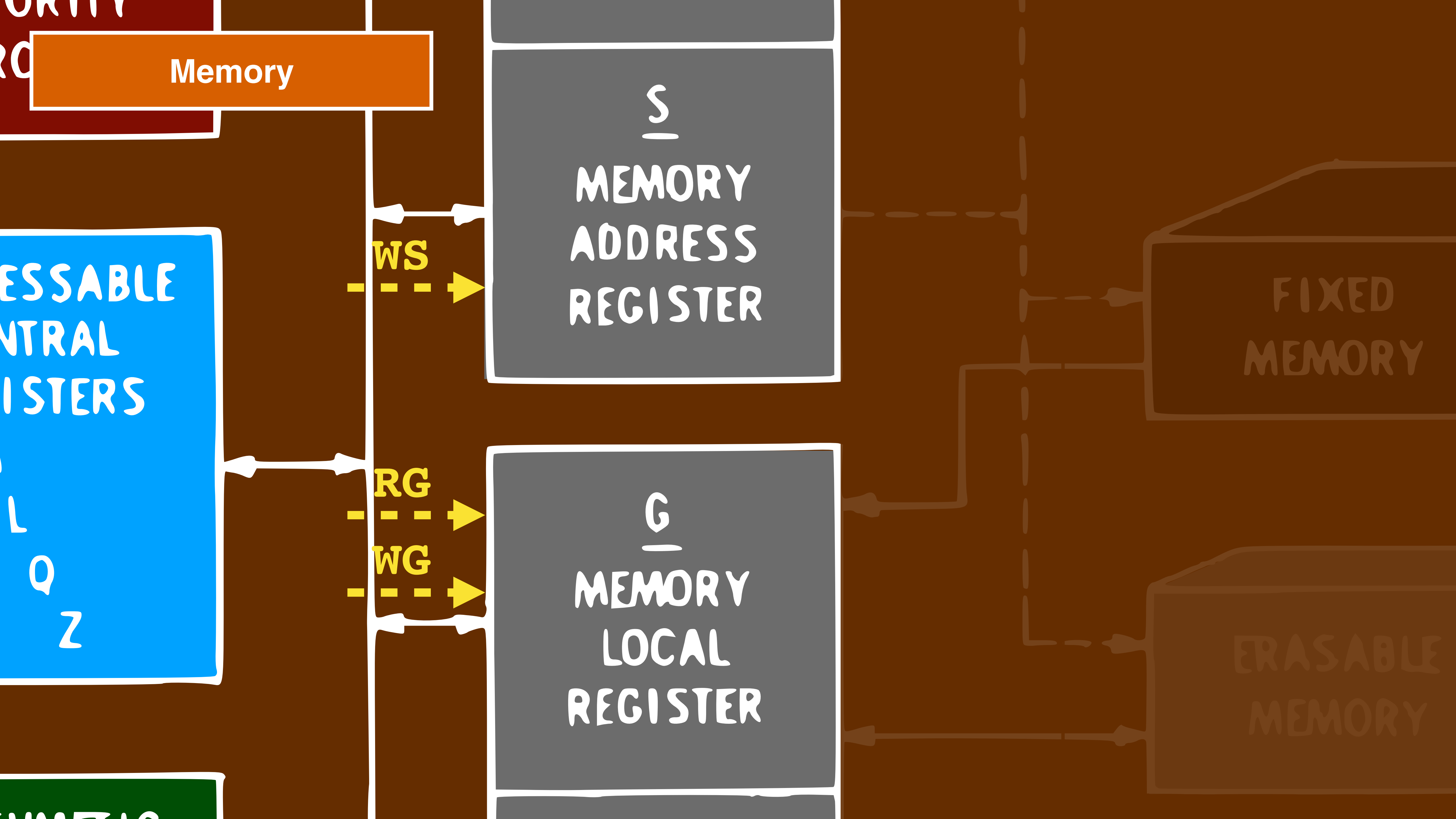
DATA PATHS

CONTROL PATHS

WRITE BUS







Memory

CLOCK  
OSCILLATOR  
AND TIMING  
PULSES

SEQUENCE  
GENERATOR  
INSTRUCTION  
MICROPROGRAM  
PULSES

MEMORY  
TIMING

INPUT  
OUTPUT

I/O COUNTERS  
AND INTERRUPT  
PRIORITY  
CIRCUITS

ADDRESSABLE  
CENTRAL  
REGISTERS  
A  
L  
Q  
Z

ARITHMETIC  
UNIT

SQ  
INSTRUCTION  
DECODER

MEMORY  
BANK  
REGISTERS

S  
MEMORY  
ADDRESS  
REGISTER

G  
MEMORY  
LOCAL  
REGISTER

PARITY

FIXED  
MEMORY

ERASABLE  
MEMORY

DATA PATHS

CONTROL PATHS

WRITE BUS

Memory

CLOCK  
OSCILLATOR  
AND TIMING  
PULSES

SEQUENCE  
GENERATOR  
INSTRUCTION  
MICROPROGRAM  
PULSES

MEMORY  
TIMING

INPUT  
OUTPUT

I/O COUNTERS  
AND INTERRUPT  
PRIORITY  
CIRCUITS

ADDRESSABLE  
CENTRAL  
REGISTERS  
A  
L  
Q  
Z

ARITHMETIC  
UNIT

SQ  
INSTRUCTION  
DECODER

MEMORY  
BANK  
REGISTERS

S  
MEMORY  
ADDRESS  
REGISTER

G  
MEMORY  
LOCAL  
REGISTER

PARITY

FIXED  
MEMORY

ERASABLE  
MEMORY

DATA PATHS

CONTROL PATHS

WRITE BUS

Id a, [k]

Microcode

T9			
T10			
T11			
T12			
T1			
T2	RSC	WG	
T3			
T4			
T5			
T6			
T7	RG	WB	
T8	RZ	WS	ST2
T9	RB	WG	
T10	RB	WA	
T11			
T12			
T1			
T2			
T3			
T4			
T5			

CLOCK  
OSCILLATOR  
AND TIMING  
PULSES

SEQUENCE  
GENERATOR  
INSTRUCTION  
MICROPROGRAM  
PULSES

MEMORY  
TIMING

INPUT  
OUTPUT

I/O COUNTERS  
AND INTERRUPT  
PRIORITY  
CIRCUITS

ADDRESSABLE  
CENTRAL  
REGISTERS  
A  
L  
Q  
Z

ARITHMETIC  
UNIT

SQ  
INSTRUCTION  
DECODER

MEMORY  
BANK  
REGISTERS

S  
MEMORY  
ADDRESS  
REGISTER

G  
MEMORY  
LOCAL  
REGISTER

PARITY

FIXED  
MEMORY

ERASABLE  
MEMORY

DATA PATHS

CONTROL PATHS

WRITE BUS



Id a, [k]

Microcode

T9			
T10			
T11			
T12			
T1			
T2	RSC	WG	
T3			
T4			
T5			
T6			
T7	RG	WB	
T8	RZ	WS	ST2
T9	RB	WG	
T10	RB	WA	
T11			
T12			
T1			
T2			
T3			
T4			
T5			

CLOCK  
OSCILLATOR  
AND TIMING  
PULSES

SEQUENCE  
GENERATOR  
INSTRUCTION  
MICROPROGRAM  
PULSES

MEMORY  
TIMING

INPUT  
OUTPUT

I/O COUNTERS  
AND INTERRUPT  
PRIORITY  
CIRCUITS

ADDRESSABLE  
CENTRAL  
REGISTERS  
A  
L  
Q  
Z

ARITHMETIC  
UNIT

SQ  
INSTRUCTION  
DECODER

MEMORY  
BANK  
REGISTERS

S  
MEMORY  
ADDRESS  
REGISTER

G  
MEMORY  
LOCAL  
REGISTER

PARITY

FIXED  
MEMORY

ERASABLE  
MEMORY

DATA PATHS

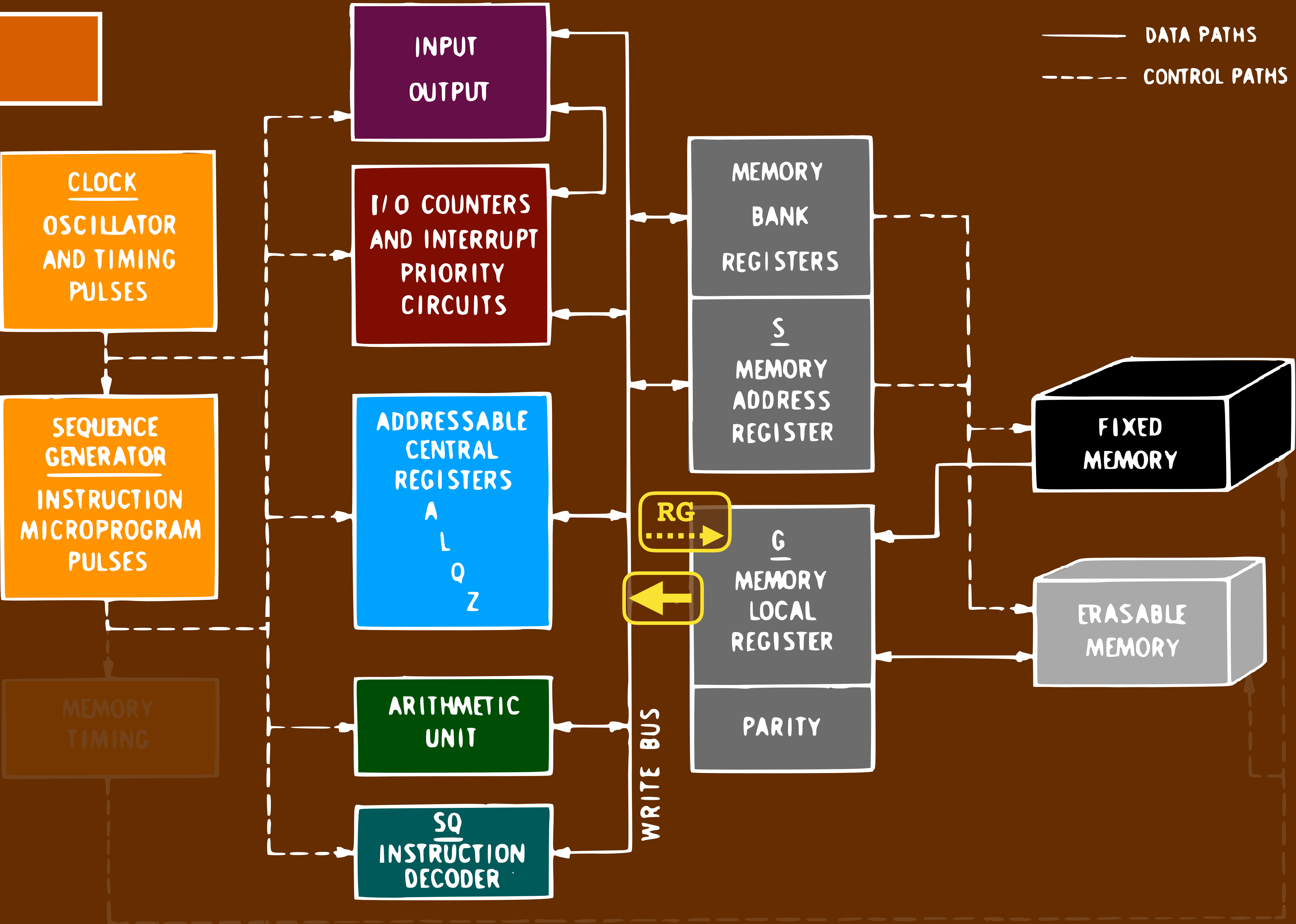
CONTROL PATHS

WRITE BUS

Id a, [k]

Microcode

T9			
T10			
T11			
T12			
T1			
T2	RSC	WG	
T3			
T4			
T5			
T6			
T7	RG	WB	
T8	RZ	WS	ST2
T9	RB	WG	
T10	RB	WA	
T11			
T12			
T1			
T2			
T3			
T4			
T5			



Id a, [k]

Microcode

T9			
T10			
T11			
T12			
T1			
T2	RSC	WG	
T3			
T4			
T5			
T6			
T7	RG	WB	
T8	RZ	WS	ST2
T9	RB	WG	
T10	RB	WA	
T11			
T12			
T1			
T2			
T3			
T4			
T5			

CLOCK  
OSCILLATOR  
AND TIMING  
PULSES

SEQUENCE  
GENERATOR  
INSTRUCTION  
MICROPROGRAM  
PULSES

MEMORY  
TIMING

INPUT  
OUTPUT

I/O COUNTERS  
AND INTERRUPT  
PRIORITY  
CIRCUITS

ADDRESSABLE  
CENTRAL  
REGISTERS  
A  
L  
Q  
Z

ARITHMETIC  
UNIT

SQ  
INSTRUCTION  
DECODER

MEMORY  
BANK  
REGISTERS

S  
MEMORY  
ADDRESS  
REGISTER

RG  
G  
MEMORY  
LOCAL  
REGISTER  
PARITY

FIXED  
MEMORY

ERASABLE  
MEMORY

DATA PATHS  
CONTROL PATHS

WRITE BUS

Id a, [k]

Microcode

T9			
T10			
T11			
T12			
T1			
T2	RSC	WG	
T3			
T4			
T5			
T6			
T7	RG	WB	
T8	RZ	WS	ST2
T9	RB	WG	
T10	RB	WA	
T11			
T12			
T1			
T2			
T3			
T4			
T5			

CLOCK  
OSCILLATOR  
AND TIMING  
PULSES

SEQUENCE  
GENERATOR  
INSTRUCTION  
MICROPROGRAM  
PULSES

MEMORY  
TIMING

INPUT  
OUTPUT

I/O COUNTERS  
AND INTERRUPT  
PRIORITY  
CIRCUITS

ADDRESSABLE  
CENTRAL  
REGISTERS  
A  
L  
Q  
Z

ARITHMETIC  
UNIT

SQ  
INSTRUCTION  
DECODER

MEMORY  
BANK  
REGISTERS

S  
MEMORY  
ADDRESS  
REGISTER

G  
MEMORY  
LOCAL  
REGISTER

PARITY

B  
BUFFER

FIXED  
MEMORY

ERASABLE  
MEMORY

DATA PATHS

CONTROL PATHS

BUS

RG

←

WB

→



Id a, [k]

T9			
T10			
T11			
T12			
T1			
T2	RSC	WG	
T3			
T4			
T5			
T6			
T7	RG	WB	
T8	RZ	WS	ST2
T9	RB	WG	
T10	RB	WA	
T11			
T12			
T1			
T2			
T3			
T4			
T5			

Microcode

CLOCK  
OSCILLATOR  
AND TIMING  
PULSES

SEQUENCE  
GENERATOR  
INSTRUCTION  
MICROPROGRAM  
PULSES

MEMORY  
TIMING

INPUT  
OUTPUT

I/O COUNTERS  
AND INTERRUPT  
PRIORITY  
CIRCUITS

ADDRESSABLE  
CENTRAL  
REGISTERS  
A  
L  
Q  
Z

ARITHMETIC  
UNIT

SQ  
INSTRUCTION  
DECODER

MEMORY  
BANK  
REGISTERS

S  
MEMORY  
ADDRESS  
REGISTER

G  
MEMORY  
LOCAL  
REGISTER

PARITY

B  
BUFFER

FIXED  
MEMORY

ERASABLE  
MEMORY

DATA PATHS

CONTROL PATHS

WRITE BUS



Id a, [k]

T9			
T10			
T11			
T12			
T1			
T2	RSC	WG	
T3			
T4			
T5			
T6			
T7	RG	WB	
T8	RZ	WS	ST2
T9	RB	WG	
T10	RB	WA	
T11			
T12			
T1			
T2			
T3			
T4			
T5			

Microcode

CLOCK  
OSCILLATOR  
AND TIMING  
PULSES

SEQUENCE  
GENERATOR  
INSTRUCTION  
MICROPROGRAM  
PULSES

MEMORY  
TIMING

INPUT  
OUTPUT

I/O COUNTERS  
AND INTERRUPT  
PRIORITY  
CIRCUITS

WA  
ADDRESSABLE  
CENTRAL  
REGISTERS  
A  
Q  
Z

ARITHMETIC  
UNIT

SQ  
INSTRUCTION  
DECODER

MEMORY  
BANK  
REGISTERS  
S  
MEMORY  
ADDRESS  
REGISTER

G  
MEMORY  
LOCAL  
REGISTER  
PARITY

B  
BUFFER

DATA PATHS  
CONTROL PATHS

FIXED  
MEMORY

ERASABLE  
MEMORY

BUS

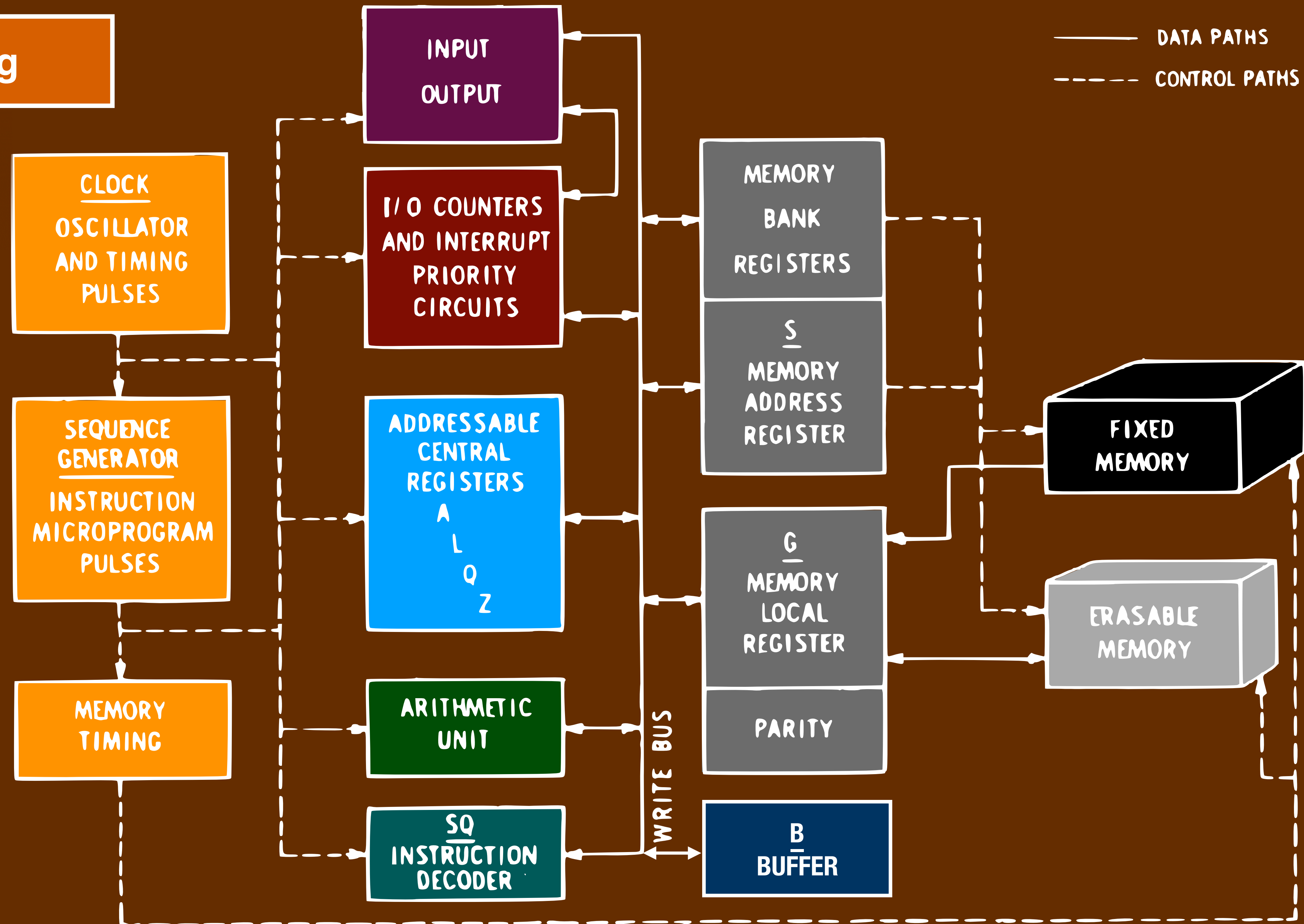
RB

←

## Memory Timing

# Id a, [k]

T9			
T10			
T11			
T12			
T1			
T2	RSC	WG	
T3			
T4			
T5			
T6			
T7	RG	WB	
T8	RZ	WS	ST2
T9	RB	WG	
T10	RB	WA	
T11			
T12			
T1			
T2			
T3			
T4			
T5			

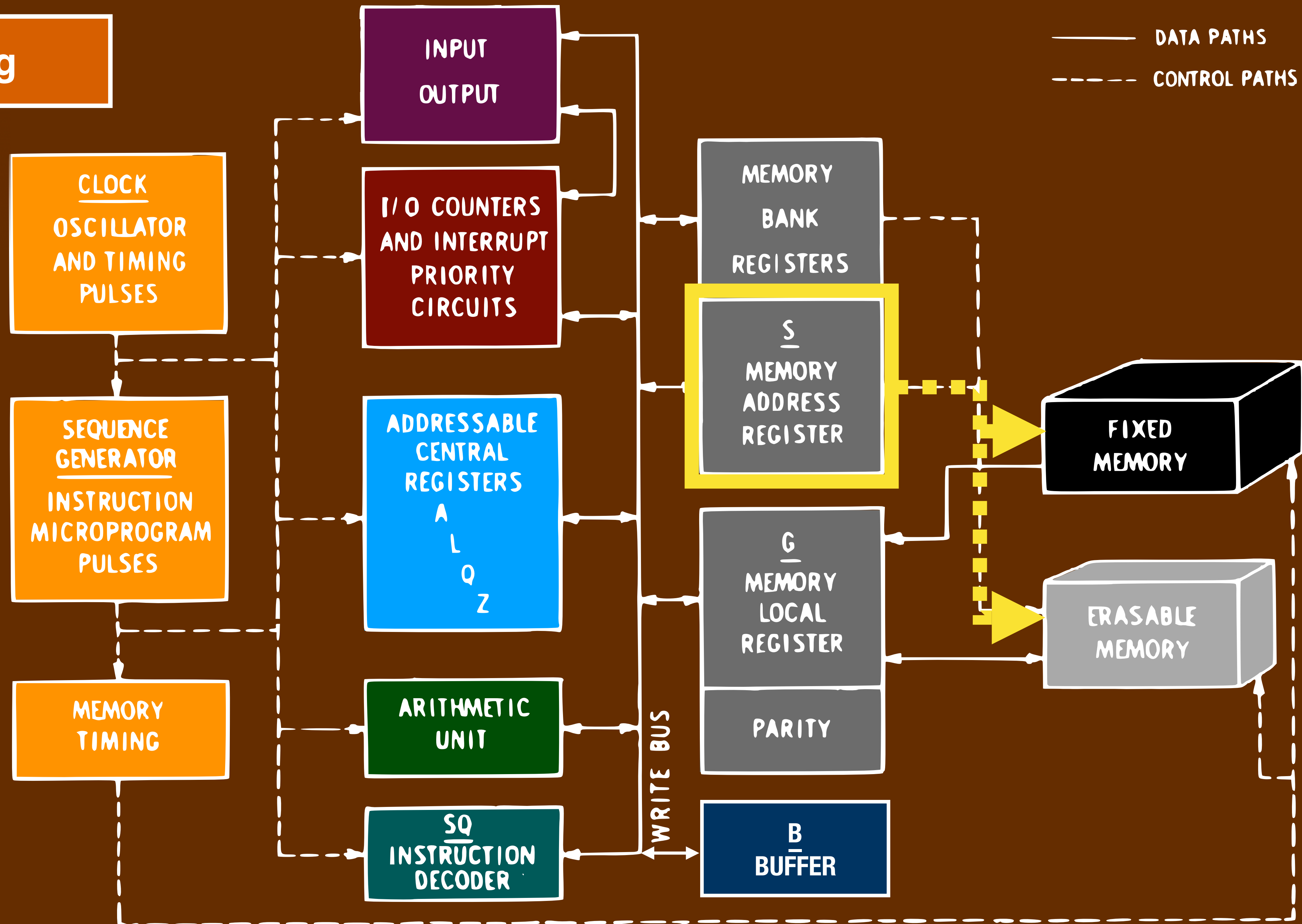


Id a, [k]

# Memory Timing

T9			
T10			
T11			
T12			
T1			
T2	RSC	WG	READ
T3			
T4			
T5			
T6			
T7	RG	WB	
T8	RZ	WS	ST2
T9	RB	WG	
T10	RB	WA	
T11			
T12			
T1			
T2			
T3			
T4			
T5			

G = M[S]

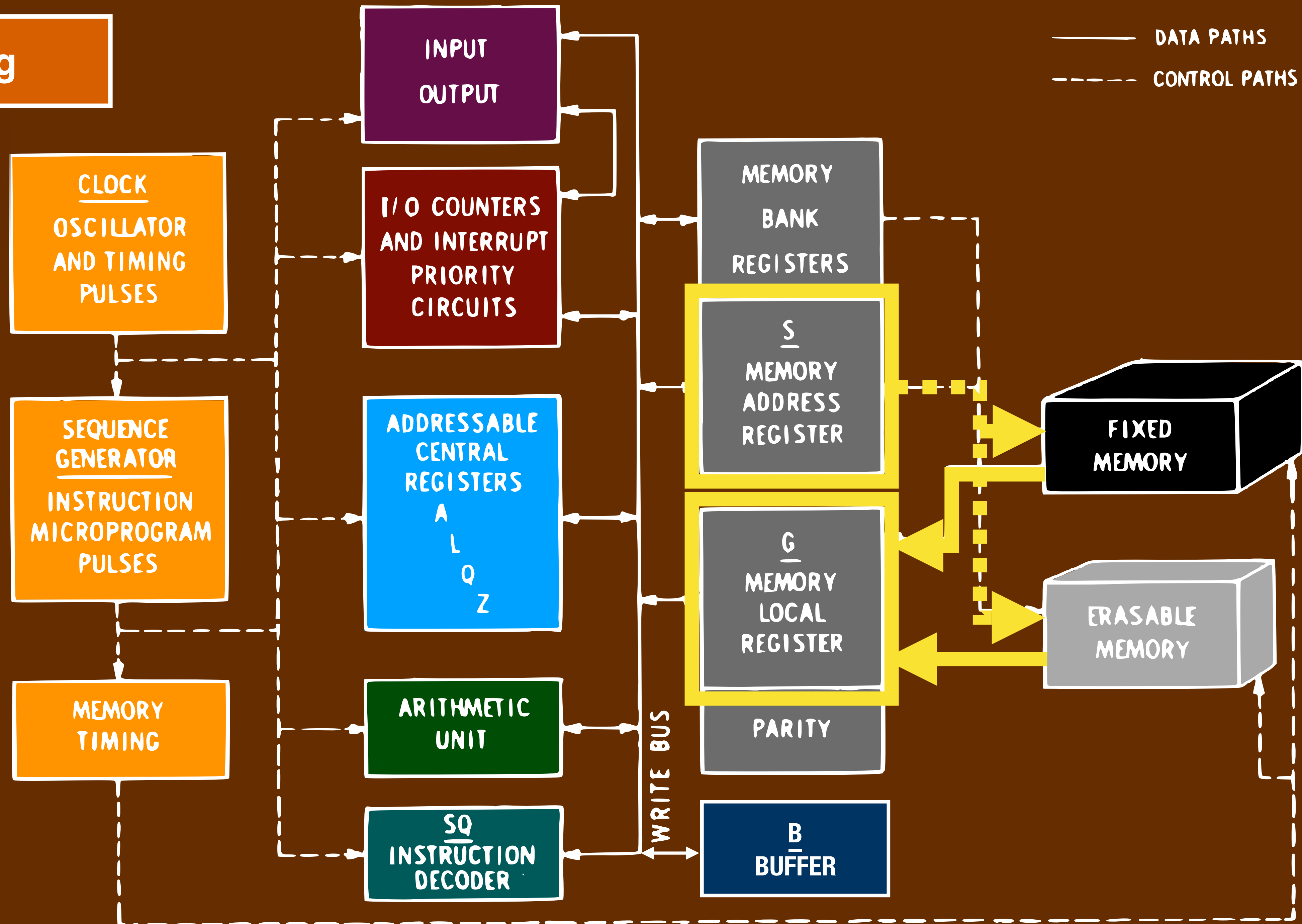


Id a, [k]

# Memory Timing

T9			
T10			
T11			
T12			
T1			
T2	RSC	WG	READ
T3			
T4			
T5			
T6			
T7	RG	WB	
T8	RZ	WS	ST2
T9	RB	WG	
T10	RB	WA	
T11			
T12			
T1			
T2			
T3			
T4			
T5			

G = M[S]



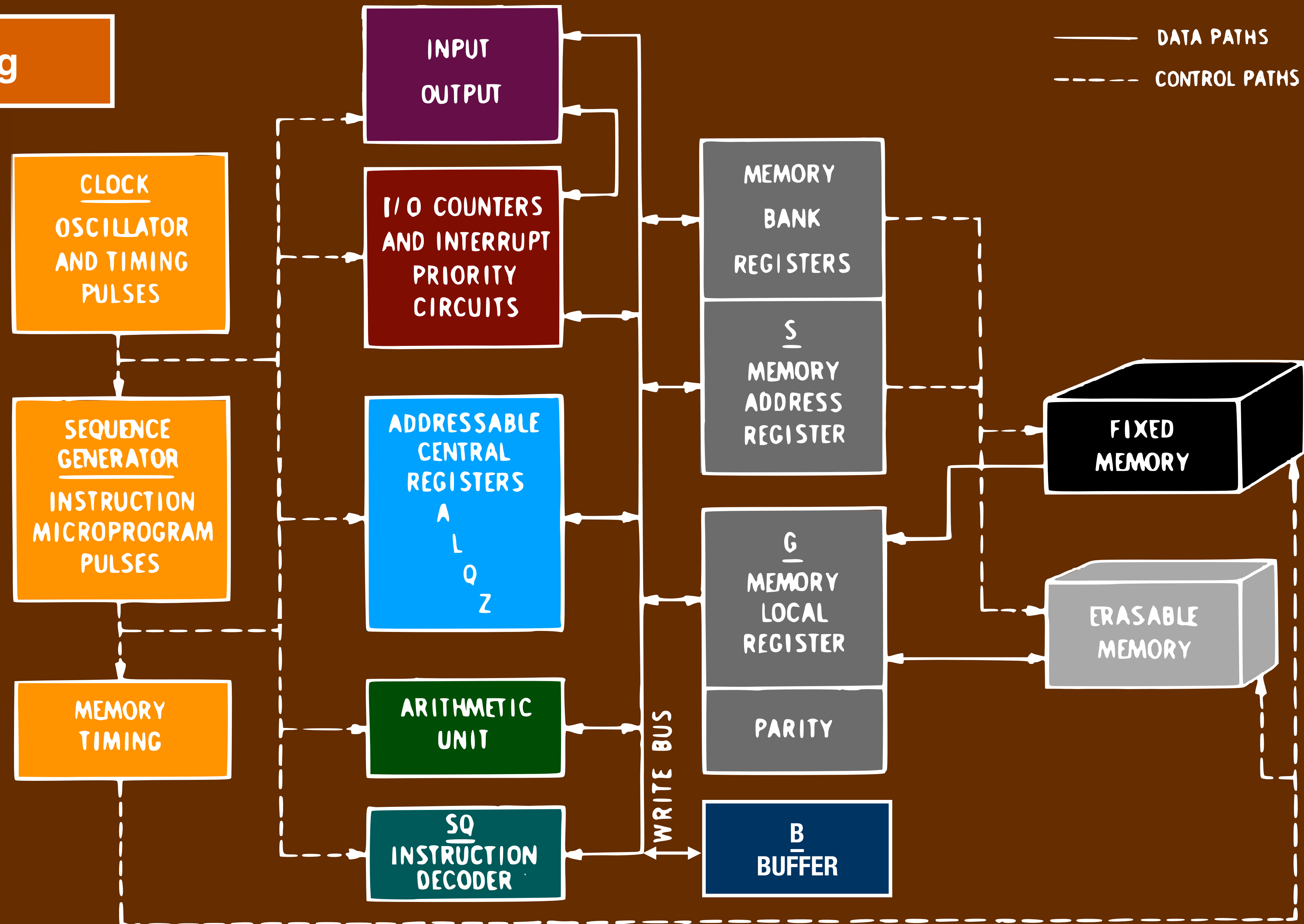


# Memory Timing

Id a, [k]

T9			
T10			
T11			
T12			
T1			
T2	RSC	WG	<div> <div>READ</div> <div>=</div> <div>GMS</div> </div>
T3			
T4			
T5			
T6			
T7	RG	WB	
T8	RZ	WS	ST2
T9	RB	WG	
T10	RB	WA	
T11			
T12			
T1			
T2			
T3			
T4			
T5			

**READ**  
**G = M[S]**

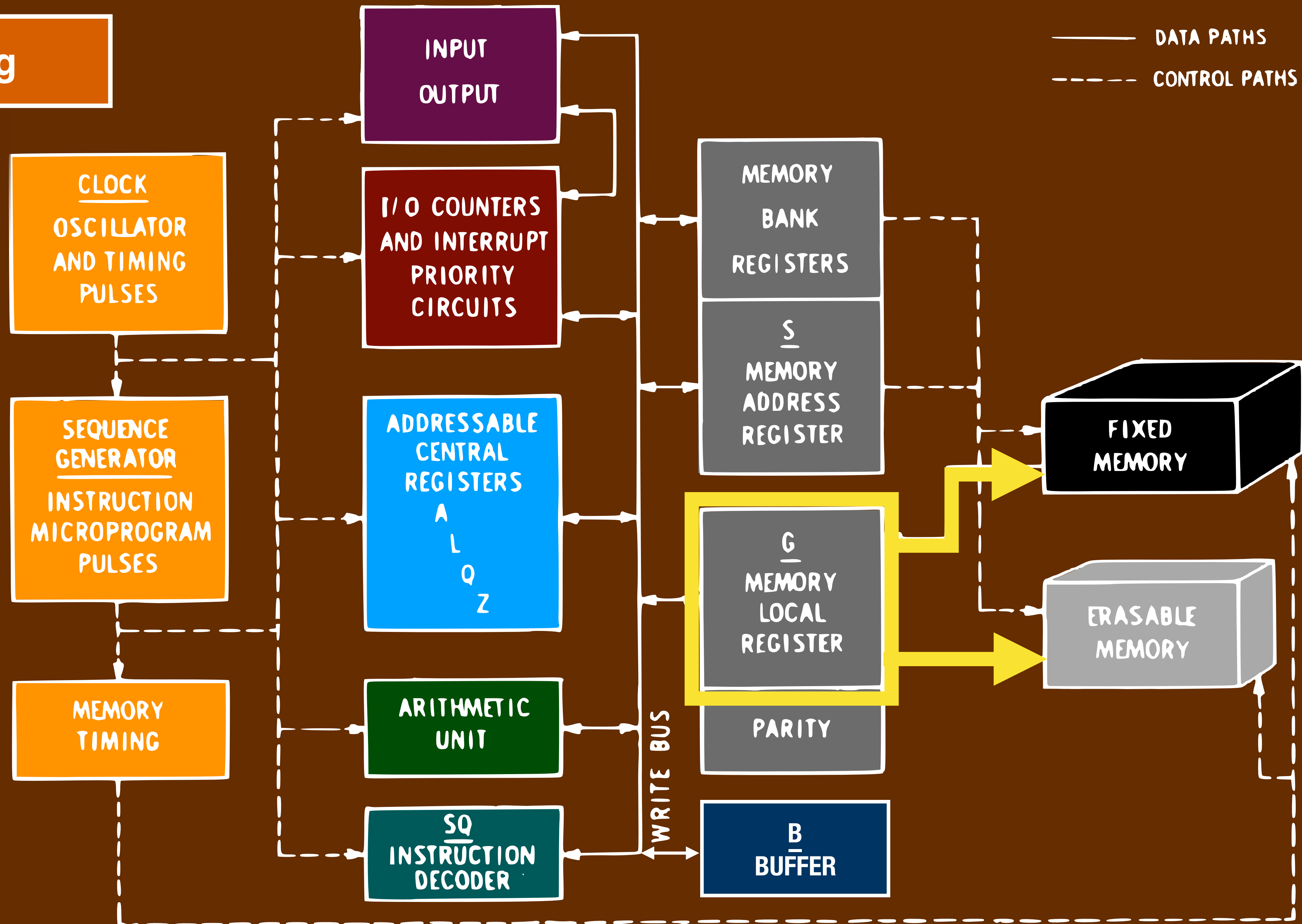




Id a, [k]

# Memory Timing

T9				
T10				
T11				
T12				
T1				
T2	RSC	WG	READ	
T3			G = M[S]	
T4				
T5				
T6				
T7	RG	WB		
T8	RZ	WS	ST2	
T9	RB	WG		
T10	RB	WA	WRITE	
T11			M[S] = G	
T12				
T1				
T2				
T3				
T4				
T5				



# Microcode

T9				
T10				
T11				
T12				
T1				
T2	RSC	WG	READ = G[M[S]]	
T3				
T4				
T5				
T6				
T7	RG	WB		
T8	RZ	WS	ST2	
T9	RB	WG		
T10	RB	WA	WRITE G =[M[S]]	
T11				
T12				
T1				
T2				
T3				
T4				
T5				

# Microcode

## Memory Timing

$$G = M[S]$$

$$M[S_{old}] = G$$

T9			
T10			
T11			
T12			
T1			
T2	RSC	WG	
T3			
T4			
T5			
T6			
T7	RG	WB	
T8	RZ	WS	ST2
T9	RB	WG	
T10	RB	WA	
T11			
T12			
T1			
T2			
T3			
T4			
T5			

Microcode

Memory  
Timing

G = M[S]

M[S<sub>old</sub>] = G

T9

T10

T11

T12

T1

T2    RSC   WG

T3

T4

T5

T6

T7    RG    WB

T8    RZ    WS   ST2

T9    RB    WG

T10   RB    WA

T11

T12

T1

T2

T3

T4

T5

ld a, [k]

G = r[S]

B = G

S = Z

G = B

A = B

Microcode

Memory  
Timing

G = M[S]

M[S<sub>old</sub>] = G

T9			
T10			
T11			
T12			
T1			
T2	RSC	WG	
T3			
T4			
T5			
T6			
T7	RG	WB	
T8	RZ	WS	ST2
T9	RB	WG	
T10	RB	WA	
T11			
T12			
T1			
T2			
T3			
T4			
T5			

ld a, [k]

G = r[S]

B = G

S = Z

G = B

A = B



# Microcode

## Memory Timing

$G = M[S]$

$M[S_{old}] = G$

T9			
T10			
T11			
T12			
T1			
T2	RSC	WG	
T3			
T4			
T5			
T6			
T7	RG	WB	
T8	RZ	WS	ST2
T9	RB	WG	
T10	RB	WA	
T11			
T12			
T1			
T2			
T3			
T4			
T5			

ld a, [k]

$$G = r[S]$$

$$B = G$$

$$S = Z$$

$$G = B$$

$$A = B$$

Microcode

Memory  
Timing

$G = M[S]$

$M[S_{old}] = G$

T9			
T10			
T11			
T12			
T1			
T2	RSC	WG	
T3			
T4			
T5			
T6			
T7	RG	WB	
T8	RZ	WS	ST2
T9	RB	WG	
T10	RB	WA	
T11			
T12			
T1			
T2			
T3			
T4			
T5			

ld a, [k]

$$G = r[S]$$

$$B = G$$

$$S = Z$$

$$G = B$$

$$A = B$$

# Microcode

## Memory Timing

$$G = M[S]$$

$$M[S_{old}] = G$$

T9			
T10			
T11			
T12			
T1			
T2	RSC	WG	READ = M[S]
T3			
T4			
T5			
T6			
T7	RG	WB	
T8	RZ	WS	ST2
T9	RB	WG	
T10	RB	WA	WRITE M[S] = G
T11			
T12			
T1			
T2			
T3			
T4			
T5			

ld a, [k]

G = r[S]

B = G

S = Z

G = B

A = B

Microcode

Memory  
Timing

G = M[S]

M[S<sub>old</sub>] = G

T9			
T10			
T11			
T12			
T1			
T2	RSC	WG	READ G = M[S]
T3			
T4			
T5			
T6			
T7	RG	WB	
T8	RZ	WS	ST2
T9	RB	WG	
T10	RB	WA	WRITE M[S] = G
T11			
T12			
T1			
T2			
T3			
T4			
T5			

ld a, [k]

G = r[S]

B = G

S = Z

G = B

A = B

T9			
T10			
T11			
T12			
T1	RL10BB	WS	
T2	RSC		WG
T3	RA		WB
T4			
T5	RG		WA
T6			
T7	RB	WSC	WG
T8	RZ	WS	ST2
T9			
T10			
T11			
T12			
T1			
T2			
T3			
T4			
T5			

xchg a, [k]

S = B[0:9]

G = r[S]

B = A

A = G

r[S] = B, G = B

S = Z

# Microcode

## Memory Timing

G = M[S]

M[S<sub>old</sub>] = G

T9			
T10			
T11			
T12			
T1			
T2	RSC	WG	READ G = M[S]
T3			
T4			
T5			
T6			
T7	RG	WB	
T8	RZ	WS	ST2
T9	RB	WG	
T10	RB	WA	WRITE M[S] = G
T11			
T12			
T1			
T2			
T3			
T4			
T5			

ld a, [k]

G = r[S]

B = G

S = Z

G = B

A = B

T9			
T10			
T11			
T12			
T1	RL10BB	WS	
T2	RSC		WG
T3	RA		WB
T4			
T5	RG		WA
T6			
T7	RB	WSC	WG
T8	RZ	WS	ST2
T9			
T10			
T11			
T12			
T1			
T2			
T3			
T4			
T5			

xchg a, [k]

S = B[0:9]

G = r[S]

B = A

A = G

r[S] = B, G = B

S = Z



Microcode

Memory  
Timing

G = M[S]

M[S<sub>old</sub>] = G

T9			
T10			
T11			
T12			
T1			
T2	RSC	WG	READ
T3			G = M[S]
T4			
T5			
T6			
T7	RG	WB	
T8	RZ	WS	ST2
T9	RB	WG	
T10	RB	WA	WRITE
T11			M[S] = G
T12			
T1			
T2			
T3			
T4			
T5			

ld a, [k]

G = r[S]

B = G

S = Z

G = B

A = B

T9			
T10			
T11			
T12			
T1	RL10BB	WS	
T2	RSC		WG
T3	RA		WB
T4			
T5	RG		WA
T6			
T7	RB	WSC	WG
T8	RZ	WS	ST2
T9			
T10			
T11			
T12			
T1			
T2			
T3			
T4			
T5			

xchg a, [k]

S = B[0:9]

G = r[S]

B = A

A = G

r[S] = B, G = B

S = Z

# Microcode

## Memory Timing

G = M[S]

M[S<sub>old</sub>] = G

T9			
T10			
T11			
T12			
T1			
T2	RSC	WG	READ
T3			G = M[S]
T4			
T5			
T6			
T7	RG	WB	
T8	RZ	WS	ST2
T9	RB	WG	
T10	RB	WA	WRITE
T11			M[S] = G
T12			
T1			
T2			
T3			
T4			
T5			

ld a, [k]

G = r[S]

B = G

S = Z

G = B

A = B

T9			
T10			
T11			
T12			
T1	RL10BB	WS	
T2	RSC		WG
T3	RA		WB
T4			
T5	RG		WA
T6			
T7	RB	WSC	WG
T8	RZ	WS	ST2
T9			
T10			
T11			
T12			
T1			
T2			
T3			
T4			
T5			

xchg a, [k]

S = B[0:9]

G = r[S]

B = A

A = G

r[S] = B, G = B

S = Z

Microcode

Memory  
Timing

G = M[S]

M[S<sub>old</sub>] = G

T9			
T10			
T11			
T12			
T1			
T2	RSC	WG	READ
T3			G =
T4			M[S]
T5			
T6			
T7	RG	WB	
T8	RZ	WS	ST2
T9	RB	WG	
T10	RB	WA	WRITE
T11			M[S]
T12			= G
T1			
T2			
T3			
T4			
T5			

ld a, [k]

G = r[S]

B = G

S = Z

G = B

A = B

T9			
T10			
T11			
T12			
T1	RL10BB	WS	
T2	RSC		WG
T3	RA		WB
T4			
T5	RG		WA
T6			
T7	RB	WSC	WG
T8	RZ	WS	ST2
T9			
T10			
T11			
T12			
T1			
T2			
T3			
T4			
T5			

xchg a, [k]

S = B[0:9]

G = r[S]

B = A

A = G

r[S] = B, G = B

S = Z

# Microcode

## Memory Timing

G = M[S]

M[S<sub>old</sub>] = G

T9			
T10			
T11			
T12			
T1			
T2	RSC	WG	READ
T3			G = M[S]
T4			
T5			
T6			
T7	RG	WB	
T8	RZ	WS	ST2
T9	RB	WG	
T10	RB	WA	WRITE
T11			M[S] = G
T12			
T1			
T2			
T3			
T4			
T5			

ld a, [k]

G = r[S]

B = G

S = Z

G = B

A = B

T9			
T10			
T11			
T12			
T1	RL10BB	WS	
T2	RSC		WG
T3	RA		WB
T4			
T5	RG		WA
T6			
T7	RB	WSC	WG
T8	RZ	WS	ST2
T9			
T10			
T11			
T12			
T1			
T2			
T3			
T4			
T5			

xchg a, [k]

S = B[0:9]

G = r[S]

B = A

A = G

r[S] = B, G = B

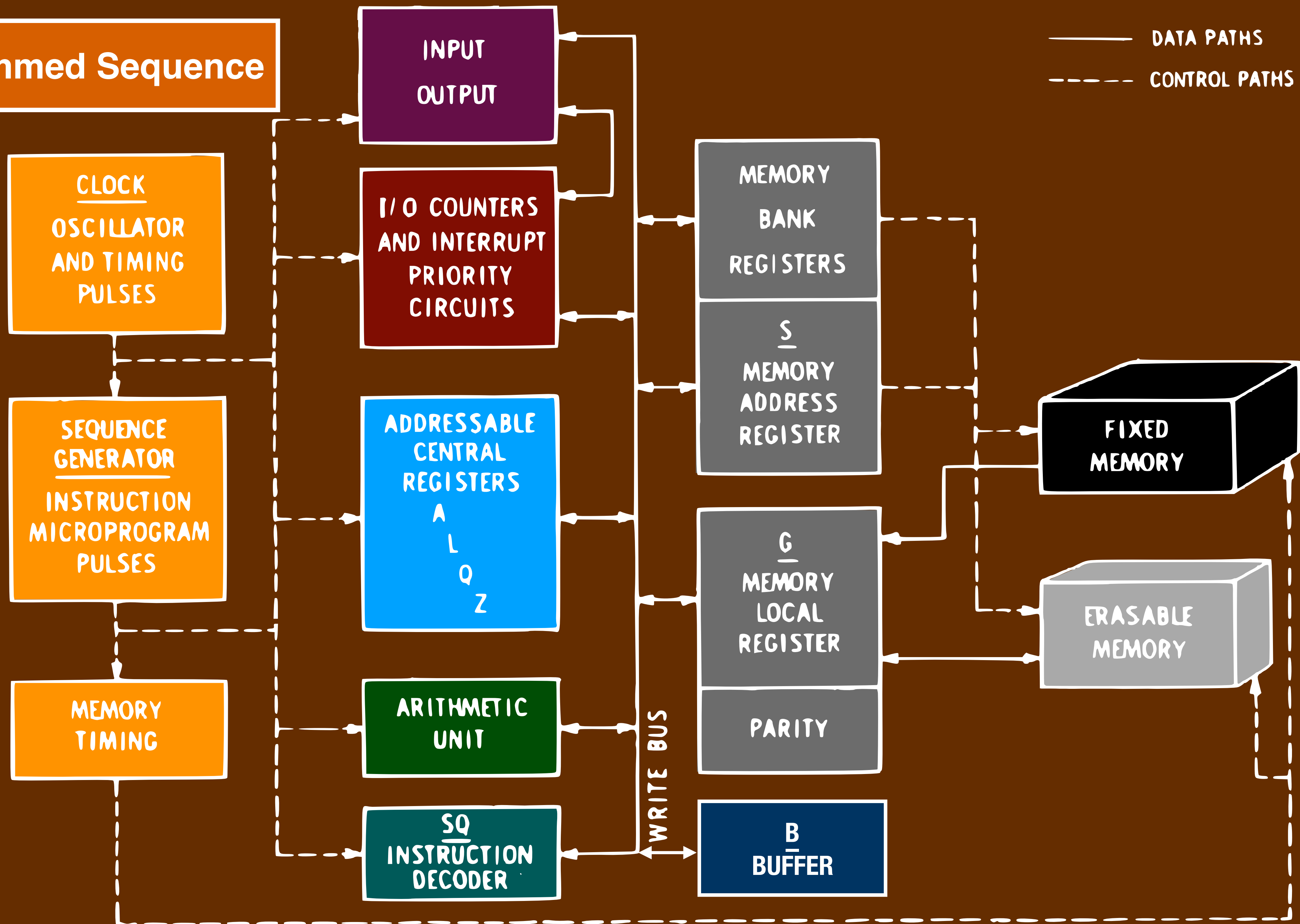
S = Z

Microcode

DV0							DV1							DV3						
1. xx	RA	WB	TSGN				4. x0	RL	WB					4. xx	L2GD	RB	WYD	A2X	PIFL	
2. 0x	RC	WA	TMZ	DVST			4. x1	RL	WB	TSGN				5. 0x	RG	WL	TSGU	DVST	CLXC	
2. 1x	DVST						5. 0x	RB	WY	B15X				5. 1x	RG	WL	TSGU	DVST	RB1F	
3. xx	RU	WB	STAGE				5. 1x	RC	WY	B15X	Z16			6. xx	RU	WB				
							6. xx	RU	WL	TOV				7. xx	L2GD	RB	WYD	A2X	PIFL	
							7. xx	RG	RSC	WB	TSGN			8. 0x	RG	WL	TSGU	DVST	CLXC	
							8. x0	RA	WY	PONEX				8. 1x	RG	WL	TSGU	DVST	RB1F	
							8. x1	RA	WY					9. xx	RU	WB				
							9. 0x	RB	WA					10. xx	L2GD	RB	WYD	A2X	PIFL	
							9. 1x	RC	WA	Z15				11. 0x	RG	WL	TSGU	DVST	CLXC	
							10. xx	RU	WB					11. 1x	RG	WL	TSGU	DVST	RB1F	
							11. xx	RL	WYD					12. xx	RU	WB				
							12. xx	RU	WL					1. xx	L2GD	RB	WYD	A2X	PIFL	
							1. xx	L2GD	RB	WYD	A2X	PIFL		2. 0x	RG	WL	TSGU	DVST	CLXC	
							2. 0x	RG	WL	TSGU	DVST	CLXC		2. 1x	RG	WL	TSGU	DVST	RB1F	
							2. 1x	RG	WL	TSGU	DVST	RB1F		3. xx	RU	WB	STAGE			
							3. xx	RU	WB	STAGE										
DV4							DV6							DV7						
3. xx	RU	WB	STAGE				4. xx	L2GD	RB	WYD	A2X	PIFL		4. xx	L2GD	RB	WYD	A2X	PIFL	
4. xx	L2GD	RB	WYD	A2X	PIFL		5. 0x	RG	WL	TSGU	DVST	CLXC		5. 0x	RG	WL	TSGU	DVST	CLXC	
5. 0x	RG	WB	TSGU	WA	CLXC		5. 1x	RG	WL	TSGU	DVST	RB1F		5. 1x	RG	WL	TSGU	DVST	RB1F	
5. 1x	RG	WB	TSGU	WA	RB1F		6. xx	RU	WB					6. xx	RU	WB				
6. xx	RZ	TOV					7. xx	L2GD	RB	WYD	A2X	PIFL		7. xx	L2GD	RB	WYD	A2X	PIFL	
7. 01	RC	WA					8. 0x	RG	WL	TSGU	DVST	CLXC		8. 0x	RG	WL	TSGU	DVST	CLXC	
7. 1x	RC	WA					8. 1x	RG	WL	TSGU	DVST	RB1F		8. 1x	RG	WL	TSGU	DVST	RB1F	
8. xx	RZ	WS	ST2	TSGN	RSTSTG		9. xx	RU	WB					9. xx	RU	WB				
9. xx	RU	WB	WL				10. xx	L2GD	RB	WYD	A2X	PIFL		10. xx	L2GD	RB	WYD	A2X	PIFL	
10. 0x	RC	WL					11. 0x	RG	WL	TSGU	DVST	CLXC		11. 0x	RG	WL	TSGU	DVST	CLXC	
							11. 1x	RG	WL	TSGU	DVST	RB1F		11. 1x	RG	WL	TSGU	DVST	RB1F	
							12. xx	RU	WB					12. xx	RU	WB				
							1. xx	L2GD	RB	WYD	A2X	PIFL		1. xx	L2GD	RB	WYD	A2X	PIFL	
							2. 0x	RG	WL	TSGU	DVST	CLXC		2. 0x	RG	WL	TSGU	DVST	CLXC	
							2. 1x	RG	WL	TSGU	DVST	RB1F		2. 1x	RG	WL	TSGU	DVST	RB1F	
							3. xx	RU	WB	STAGE				3. xx	RU	WB	STAGE			



# Unprogrammed Sequence



Unprogrammed Sequence

INPUT  
OUTPUT

CLOCK  
OSCILLATOR  
AND TIMING  
PULSES

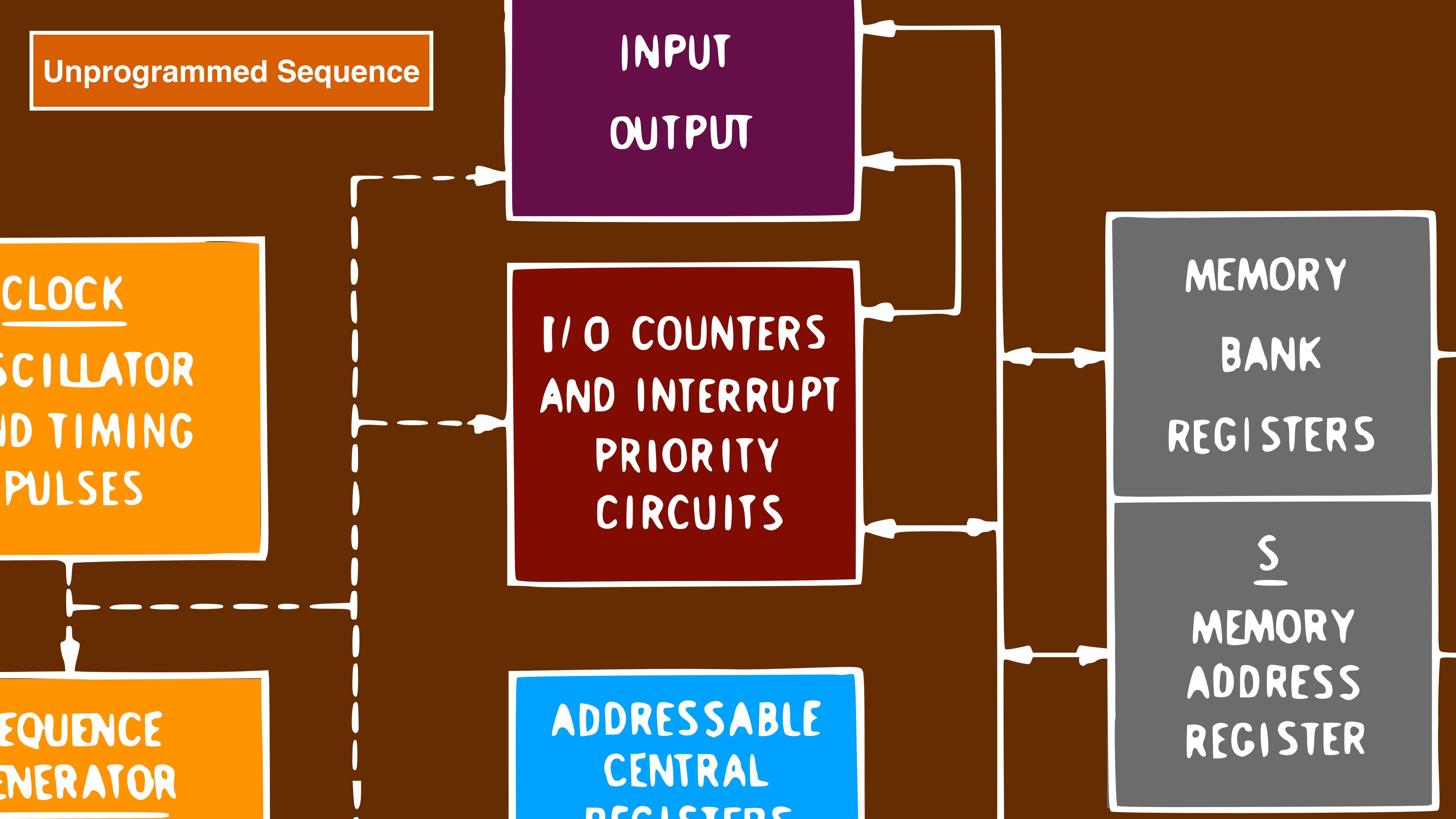
I/O COUNTERS  
AND INTERRUPT  
PRIORITY  
CIRCUITS

SEQUENCE  
GENERATOR

ADDRESSABLE  
CENTRAL  
REGISTERS

MEMORY  
BANK  
REGISTERS

S  
MEMORY  
ADDRESS  
REGISTER



Unprogrammed Sequence

INPUT  
OUTPUT

CLOCK  
OSCILLATOR  
AND TIMING  
PULSES

I/O COUNTERS  
AND INTERRUPT  
PRIORITY  
CIRCUITS

SEQUENCE  
GENERATOR

ADDRESSABLE  
CENTRAL  
REGISTERS

0000	01
0000	012
0000	013
0000	014
0000	015
0000	016
0000	017
0000	018
0000	019
0000	01A
0000	01B
0000	01C

Unprogrammed Sequence

INPUT  
OUTPUT

Increment  
TIME1

I/O COUNTERS  
AND INTERRUPT  
PRIORITY  
CIRCUITS

ADDRESSABLE  
CENTRAL  
REGISTERS

CLOCK  
OSCILLATOR  
AND TIMING  
PULSES

SEQUENCE  
GENERATOR

0000	01
0000	012
0000	013
0000	014
0000	015
0000	016
0000	017
0000	018
0000	019
0000	01A
0000	01B
0000	01C



Unprogrammed Sequence

INPUT  
OUTPUT

CLOCK  
OSCILLATOR  
AND TIMING  
PULSES

I/O COUNTERS  
AND INTERRUPT  
PRIORITY  
CIRCUITS

PINC \$015

SEQUENCE  
GENERATOR

ADDRESSABLE  
CENTRAL  
REGISTERS

Increment  
TIME1

0000	01
0000	012
0000	013
0000	014
0000	015
0000	016
0000	017
0000	018
0000	019
0000	01A
0000	01B
0000	01C



Unprogrammed Sequence

PINC  
DINC  
MINC  
MCDU  
PCDU  
SHINC  
SHANC

INOTRD  
INORLD  
FETCH  
STORE  
TCSAJ  
  
RUPT  
GOJ

INPUT  
OUTPUT

I/O COUNTERS  
AND INTERRUPT  
PRIORITY  
CIRCUITS

PINC \$015

SEQUENCE  
GENERATOR

ADDRESSABLE  
CENTRAL  
REGISTERS

Increment  
TIME1

CHIN	0000	01
ROL	0000	012
SHR8	0000	013
TIME2	0000	014
TIME1	0000	015
TIME3	0000	016
TIME4	0000	017
TIME5	0000	018
TIME6	0000	019
CDUX	0000	01A
CDUY	0000	01B
CDUZ	0000	01C

Unprogrammed Sequence

PINC  
DINC  
MINC  
MCDU  
PCDU  
SHINC  
SHANC

INOTRD  
INORLD  
FETCH  
STORE  
TCSAJ  
  
RUPT  
GOJ

INPUT  
OUTPUT

I/O COUNTERS  
AND INTERRUPT  
PRIORITY  
CIRCUITS

PINC \$015

SEQUENCE  
GENERATOR

ADDRESSABLE  
CENTRAL  
REGISTERS

Increment  
TIME1

CHIN	0000	01
ROL	0000	012
SHR8	0000	013
TIME2	0000	014
TIME1	0000	015
TIME3	0000	016
TIME4	0000	017
TIME5	0000	018
TIME6	0000	019
CDUX	0000	01A
CDUY	0000	01B
CDUZ	0000	01C

Unprogrammed Sequence

PINC  
DINC  
MINC  
MCDU  
PCDU  
SHINC  
SHANC

INOTRD  
INORLD  
FETCH  
STORE  
TCSAJ  
  
RUPT  
GOJ

INPUT  
OUTPUT

I/O COUNTERS  
AND INTERRUPT  
PRIORITY  
CIRCUITS

PINC \$015

SEQUENCE  
GENERATOR

ADDRESSABLE  
CENTRAL  
REGISTERS

Increment  
TIME1

CHIN	0000	01
ROL	0000	012
SHR8	0000	013
TIME2	0000	014
TIME1	0000	015
TIME3	0000	016
TIME4	0000	017
TIME5	0000	018
TIME6	0000	019
CDUX	0000	01A
CDUY	0000	01B
CDUZ	0000	01C



Unprogrammed Sequence

PINC  
DINC  
MINC  
MCDU  
PCDU  
SHINC  
SHANC

INOTRD  
INORLD  
FETCH  
STORE  
TCSAJ  
  
RUPT  
GOJ

INPUT  
OUTPUT

I/O COUNTERS  
AND INTERRUPT  
PRIORITY  
CIRCUITS

PINC \$015

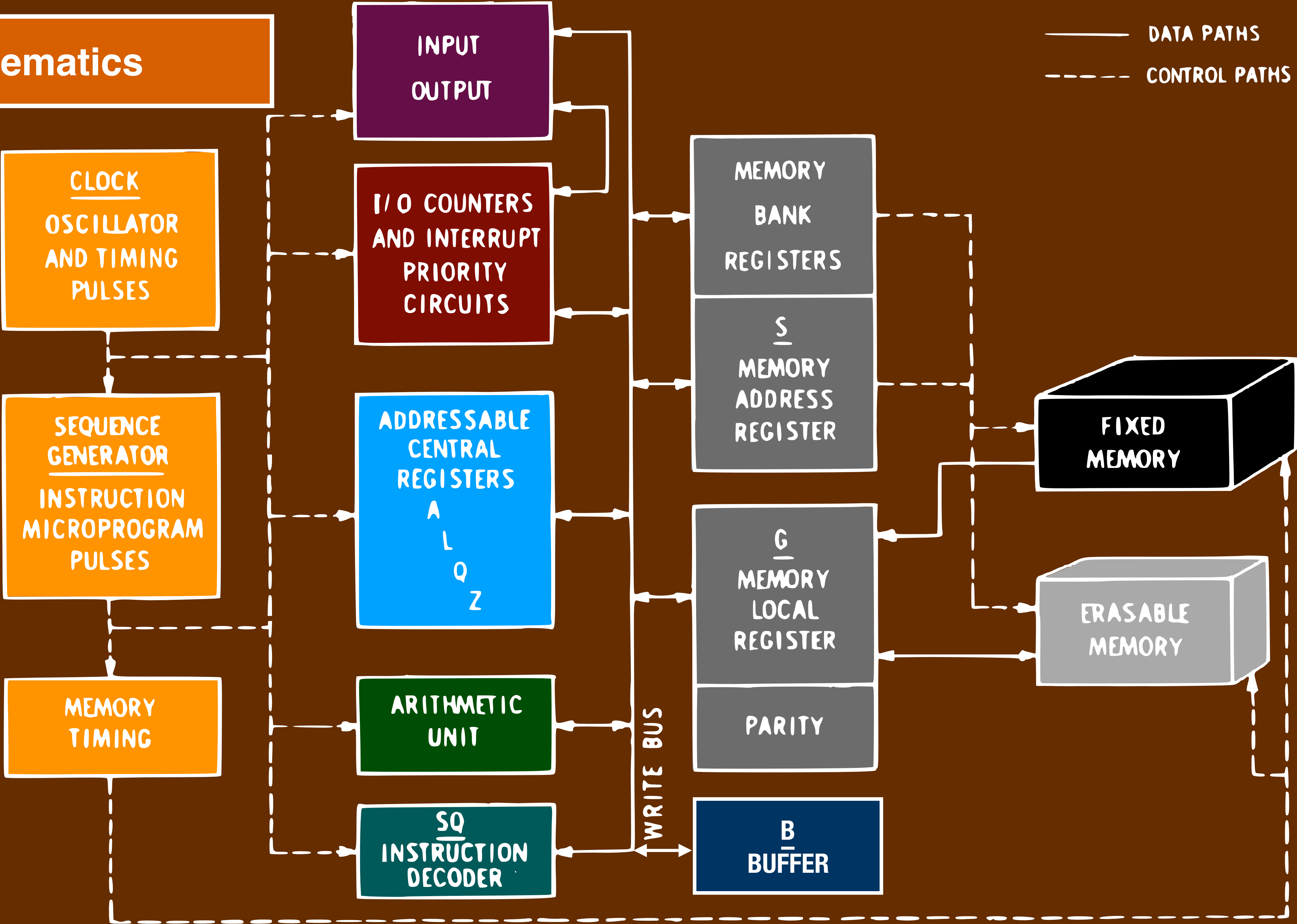
ADDRESSABLE  
CENTRAL  
REGISTERS

SEQUENCE  
GENERATOR

Increment  
TIME1

CHIN	0000	01
ROL	0000	012
SHR8	0000	013
TIME2	0000	014
TIME1	0000	015
TIME3	0000	016
TIME4	0000	017
TIME5	0000	018
TIME6	0000	019
CDUX	0000	01A
CDUY	0000	01B
CDUZ	0000	01C

# Schematics





# Schematics





# Schematics





# Schematics





A8-11

4219-4

4218-3

A8-11

4217-5

4216-2

A8-11

4221-1

4220-1

NASA NO 2003121-081 X

MFD BY RAYTHEON CO  
S/N RAY 72

LOGIC MODULE A8+H

NASA NO 2003121-091 X

MFD BY RAYTHEON CO  
S/N RAY 20

LOGIC MODULE A12

NASA NO 2003121-061 X

MFD BY RAYTHEON CO  
S/N RAY 20

LOGIC MODULE A6

NASA NO 2003121-051 X

MFD BY RAYTHEON CO  
S/N RAY 21

LOGIC MODULE A5

A04

4207-2

4206-5

A03

4205-2

4204-3

NASA NOx2003121-021

MFD BY RAYTHEON CO  
S/N RAY P 001

LOGIC MODULE A2





NASA NO 2003007-011  
REV A  
MFD BY RAYTHEON CO  
S/N 10011  
INTERFACE MODULE A27-20  
ENGINEERING PROTOTYPE

NASA NO 2003007-011  
REV A  
MFD BY RAYTHEON CO  
S/N 10011  
INTERFACE MODULE A27-20  
ENGINEERING PROTOTYPE

NASA NO 2003007-011  
REV A  
MFD BY RAYTHEON CO  
S/N 10011  
INTERFACE MODULE A27-20  
ENGINEERING PROTOTYPE

NASA NO 2003070-011  
REV A1  
MFD BY RAYTHEON CO  
S/N 10011  
INTERFACE MODULE A25-26  
ENGINEERING PROTOTYPE

NASA NO X2003070-0121  
REV C  
MFD BY RAYTHEON CO  
S/N 01Y0007  
INTERFACE MODULE A25-26

A24

A23 4243-2 4242-3

A22 4241-1 4240-3

A21 4239-2 4238-3

A20 4237-4 4236-2

A19 4235-3 4234-1

A18 4233-1 4232-1

A17 4231-1 4230-3

A16 4229-2 4228-2



A15 4227-4 4226-3

A14 4225-4 4224-3

96-40-8 808 80 27-8

48-11 4219-4 4218-3

48-11 4217-3 4216-2

48-11 4221-1 4220-1

NASA NO 2003121-011 X  
MFD BY RAYTHEON CO  
S/N RAY 72  
LOGIC MODULE A8-11

NASA NO 2003121-011 X  
MFD BY RAYTHEON CO  
S/N RAY 20  
LOGIC MODULE A12

NASA NO 2003121-011 X  
MFD BY RAYTHEON CO  
S/N RAY 20  
LOGIC MODULE A6

NASA NO 2003121-011 X  
MFD BY RAYTHEON CO  
S/N RAY 20  
LOGIC MODULE A5

A04 4207-2 4206-5

A03 4205-2 4204-3

NASA NO X2003121-021  
MFD BY RAYTHEON CO  
S/N RAY P 001  
LOGIC MODULE A2



This is the one here in the Computer History Museum.  
A12 sits in the wrong slot, and A1 and A7 are missing. :(

INTERFACE MODULE A27-20

INTERFACE MODULE A27-20

INTERFACE MODULE A27-20

INTERFACE MODULE A25-26

INTERFACE MODULE A25-26

A24 A24 INOUT VII

A23 A23 INOUT VI

A22 A22 INOUT V

A21 A21 COUNTER CELL II

A20 A20 COUNTER CELL I

A19 A19 INOUT IV

A18 A18 INOUT III

A17 A17 INOUT II

A16 A16 INOUT I

A15 RUPT SERVICE

A14 MEMORY TIMING & ADDRESSING

A13 ALARMS

A12 PARITY AND S REGISTER

A11 4 BIT MODULE

A10 4 BIT MODULE

A9 4 BIT MODULE

A8 4 BIT MODULE

A7 SERVICE GATES

A6 CROSS POINT GENERATOR II

A5 CROSS POINT GENERATOR NQI

A4 STAGE BRANCH DECODING

A3 SQ REGISTER AND DECODING

A2 TIMER

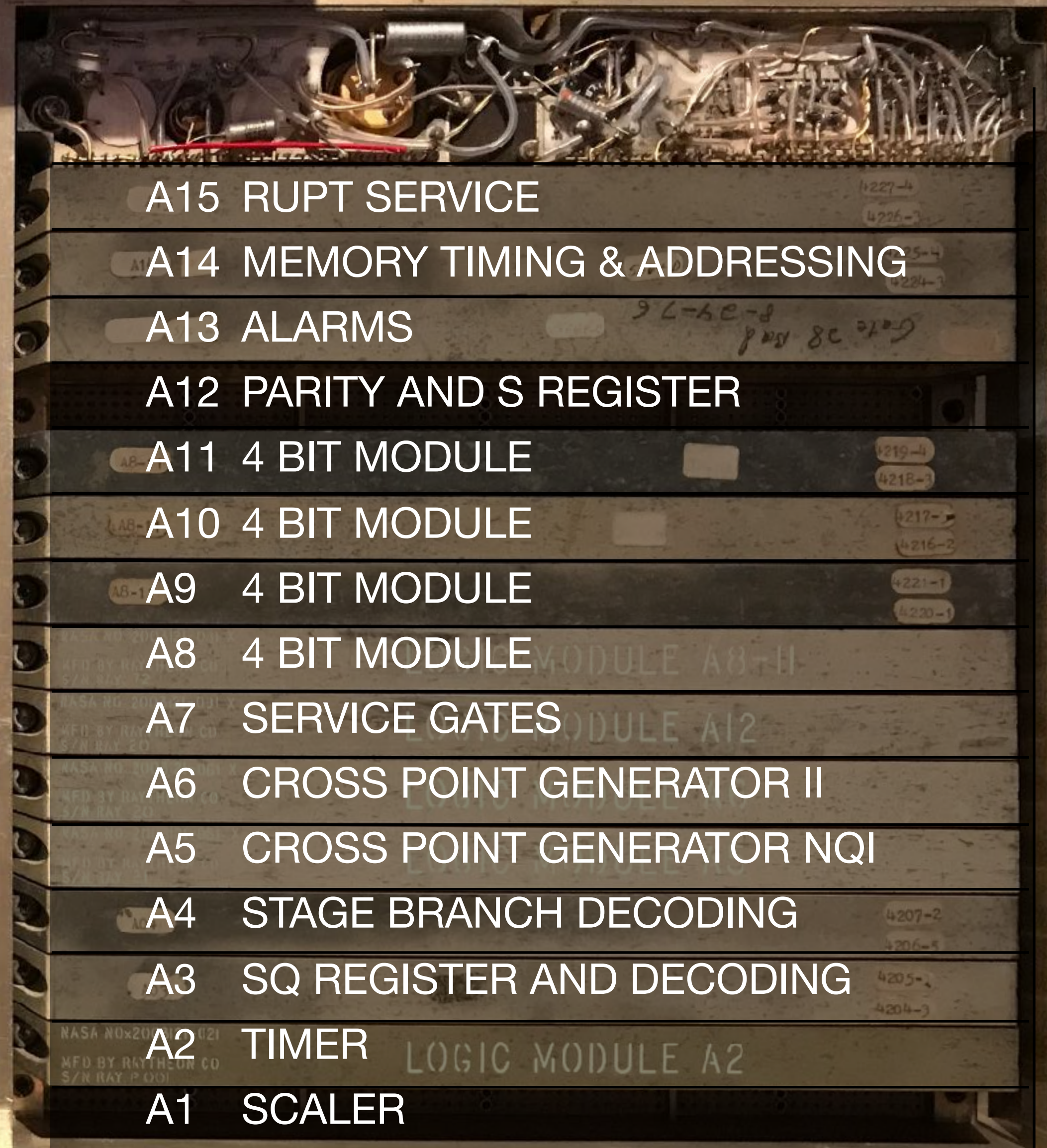
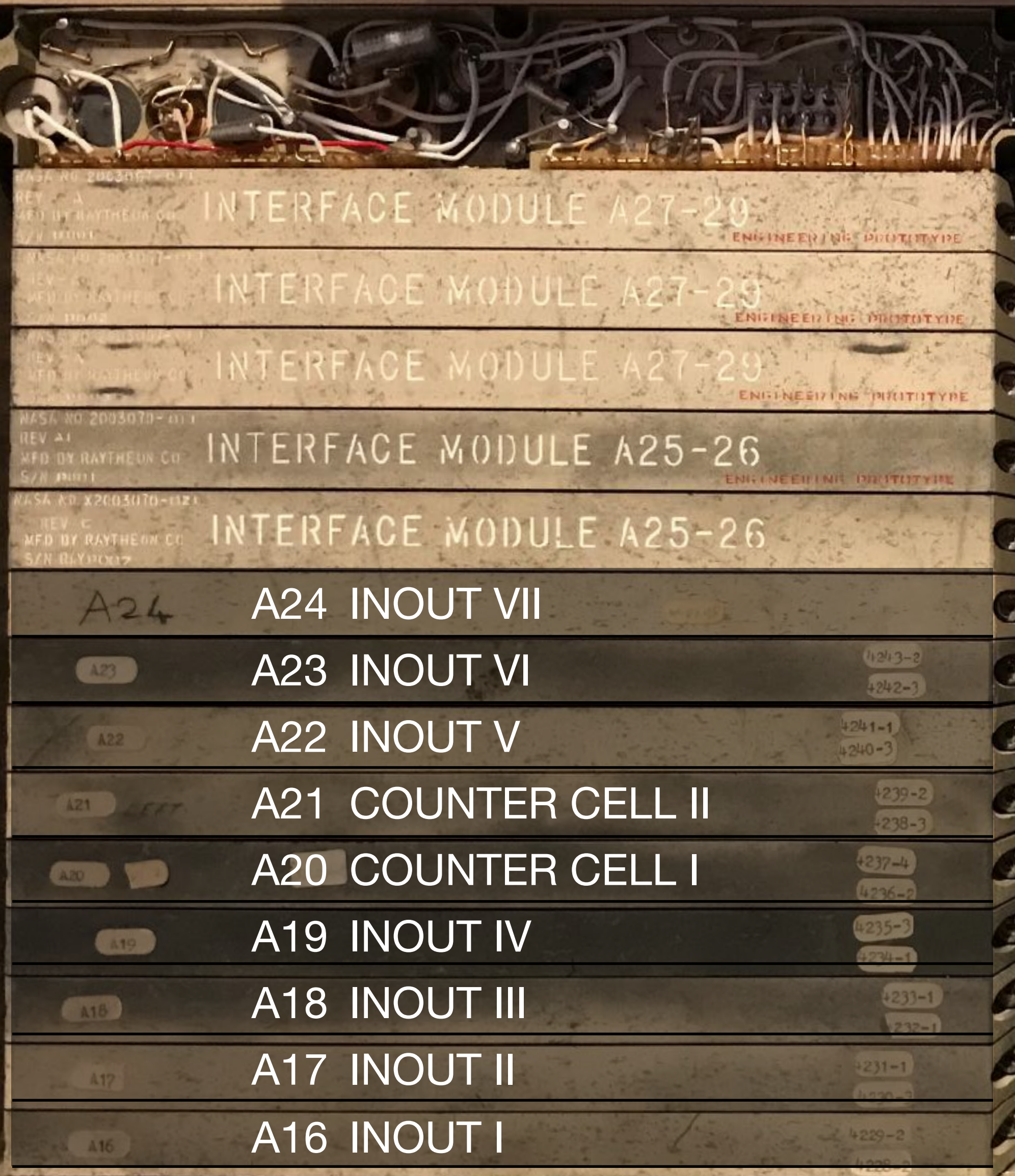
A1 SCALER



This is the one here in the Computer History Museum.  
A12 sits in the wrong slot, and A1 and A7 are missing. :(

I/O Connector

Debug





1997, 1998, 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022, 2023, 2024, 2025, 2026, 2027, 2028, 2029, 2030, 2031, 2032, 2033, 2034, 2035, 2036, 2037, 2038, 2039, 2040, 2041, 2042, 2043, 2044, 2045, 2046, 2047, 2048, 2049, 2050, 2051, 2052, 2053, 2054, 2055, 2056, 2057, 2058, 2059, 2060, 2061, 2062, 2063, 2064, 2065, 2066, 2067, 2068, 2069, 2070, 2071, 2072, 2073, 2074, 2075, 2076, 2077, 2078, 2079, 2080, 2081, 2082, 2083, 2084, 2085, 2086, 2087, 2088, 2089, 2090, 2091, 2092, 2093, 2094, 2095, 2096, 2097, 2098, 2099, 2100, 2101, 2102, 2103, 2104, 2105, 2106, 2107, 2108, 2109, 2110, 2111, 2112, 2113, 2114, 2115, 2116, 2117, 2118, 2119, 2120, 2121, 2122, 2123, 2124, 2125, 2126, 2127, 2128, 2129, 2130, 2131, 2132, 2133, 2134, 2135, 2136, 2137, 2138, 2139, 2140, 2141, 2142, 2143, 2144, 2145, 2146, 2147, 2148, 2149, 2150, 2151, 2152, 2153, 2154, 2155, 2156, 2157, 2158, 2159, 2160, 2161, 2162, 2163, 2164, 2165, 2166, 2167, 2168, 2169, 2170, 2171, 2172, 2173, 2174, 2175, 2176, 2177, 2178, 2179, 2180, 2181, 2182, 2183, 2184, 2185, 2186, 2187, 2188, 2189, 2190, 2191, 2192, 2193, 2194, 2195, 2196, 2197, 2198, 2199, 2200, 2201, 2202, 2203, 2204, 2205, 2206, 2207, 2208, 2209, 2210, 2211, 2212, 2213, 2214, 2215, 2216, 2217, 2218, 2219, 2220, 2221, 2222, 2223, 2224, 2225, 2226, 2227, 2228, 2229, 2230, 2231, 2232, 2233, 2234, 2235, 2236, 2237, 2238, 2239, 2240, 2241, 2242, 2243, 2244, 2245, 2246, 2247, 2248, 2249, 2250, 2251, 2252, 2253, 2254, 2255, 2256, 2257, 2258, 2259, 2260, 2261, 2262, 2263, 2264, 2265, 2266, 2267, 2268, 2269, 2270, 2271, 2272, 2273, 2274, 2275, 2276, 2277, 2278, 2279, 2280, 2281, 2282, 2283, 2284, 2285, 2286, 2287, 2288, 2289, 2290, 2291, 2292, 2293, 2294, 2295, 2296, 2297, 2298, 2299, 2300, 2301, 2302, 2303, 2304, 2305, 2306, 2307, 2308, 2309, 2310, 2311, 2312, 2313, 2314, 2315, 2316, 2317, 2318, 2319, 2320, 2321, 2322, 2323, 2324, 2325, 2326, 2327, 2328, 2329, 2330, 2331, 2332, 2333, 2334, 2335, 2336, 2337, 2338, 2339, 2340, 2341, 2342, 2343, 2344, 2345, 2346, 2347, 2348, 2349, 2350, 2351, 2352, 2353, 2354, 2355, 2356, 2357, 2358, 2359, 2360, 2361, 2362, 2363, 2364, 2365, 2366, 2367, 2368, 2369, 2370, 2371, 2372, 2373, 2374, 2375, 2376, 2377, 2378, 2379, 2380, 2381, 2382, 2383, 2384, 2385, 2386, 2387, 2388, 2389, 2390, 2391, 2392, 2393, 2394, 2395, 2396, 2397, 2398, 2399, 2400, 2401, 2402, 2403, 2404, 2405, 2406, 2407, 2408, 2409, 2410, 2411, 2412, 2413, 2414, 2415, 2416, 2417, 2418, 2419, 2420, 2421, 2422, 2423, 2424, 2425, 2426, 2427, 2428, 2429, 2430, 2431, 2432, 2433, 2434, 2435, 2436, 2437, 2438, 2439, 2440, 2441, 2442, 2443, 2444, 2445, 2446, 2447, 2448, 2449, 2450, 2451, 2452, 2453, 2454, 2455, 2456, 2457, 2458, 2459, 2460, 2461, 2462, 2463, 2464, 2465, 2466, 2467, 2468, 2469, 2470, 2471, 2472, 2473, 2474, 2475, 2476, 2477, 2478, 2479, 2480, 2481, 2482, 2483, 2484, 2485, 2486, 2487, 2488, 2489, 2490, 2491, 2492, 2493, 2494, 2495, 2496, 2497, 2498, 2499, 2500, 2501, 2502, 2503, 2504, 2505, 2506, 2507, 2508, 2509, 2510, 2511, 2512, 2513, 2514, 2515, 2516, 2517, 2518, 2519, 2520, 2521, 2522, 2523, 2524, 2525, 2526, 2527, 2528, 2529, 2530, 2531, 2532, 2533, 2534, 2535, 2536, 2537, 2538, 2539, 2540, 2541, 2542, 2543, 2544, 2545, 2546, 2547, 2548, 2549, 2550, 2551, 2552, 2553, 2554, 2555, 2556, 2557, 2558, 2559, 2560, 2561, 2562, 2563, 2564, 2565, 2566, 2567, 2568, 2569, 2570, 2571, 2572, 2573, 2574, 2575, 2576, 2577, 2578, 2579, 2580, 2581, 2582, 2583, 2584, 2585, 2586, 2587, 2588, 2589, 2590, 2591, 2592, 2593, 2594, 2595, 2596, 2597, 2598, 2599, 2600, 2601, 2602, 2603, 2604, 2605, 2606, 2607, 2608, 2609, 2610, 2611, 2612, 2613, 2614, 2615, 2616, 2617, 2618, 2619, 2620, 2621, 2622, 2623, 2624, 2625, 2626, 2627, 2628, 2629, 2630, 2631, 2632, 2633, 2634, 2635, 2636, 2637, 2638, 2639, 2640, 2641, 2642, 2643, 2644, 2645, 2646, 2647, 2648, 2649, 2650, 2651, 2652, 2653, 2654, 2655, 2656, 2657, 2658, 2659, 2660, 2661, 2662, 2663, 2664, 2665, 2666, 2667, 2668, 2669, 2670, 2671, 2672, 2673, 2674, 2675, 2676, 2677, 2678, 26

NAME: FU JINGSHI-CH  
 REV 6.1  
 AFD BY: JAVIER-RODRIGUEZ  
 DATE: MAY 19 1991

DATA NO. 000100  
REV. 1  
APD BY RAYTHEON CO  
E/A RAY 0001

WASA #0 2003026-08  
REV A,  
REF BT BAYTHORN CO  
LFA BAYTHORN

WALL NO. 2001007-01  
REV A.  
REF BY HAYTHORN CO  
SUN JAY, ALABAMA

MAILED BY COMMUNICATIONS  
DIVISION  
RECEIVED BY MATTHEW H. C.  
JAN 11 1961

WASH. DC. (UPI) - The  
U.S. Coast Guard  
said it has received  
a report of a  
sinking ship.

WFO OF DAYTON CO  
2/20 1964

1

# ENGINEERING PROTOTYPE

ENGINEERING  
PROSTITUTE

Q 24 1st  
4 June 1944



OSCILLATOR MODULE B7

ALARM MODULE B8

ERASABLE DRIVER MODULE B9-10

ERASABLE DRIVER MODULE B9-10

CURRENT SWITCH MODULE B11

RAM

SENSE AMPLIFIER MODULE B13-14

SENSE AMPLIFIER MODULE B13-14

STRAND SELECT MODULE B15

ROPE DRIVER MODULE B16-17

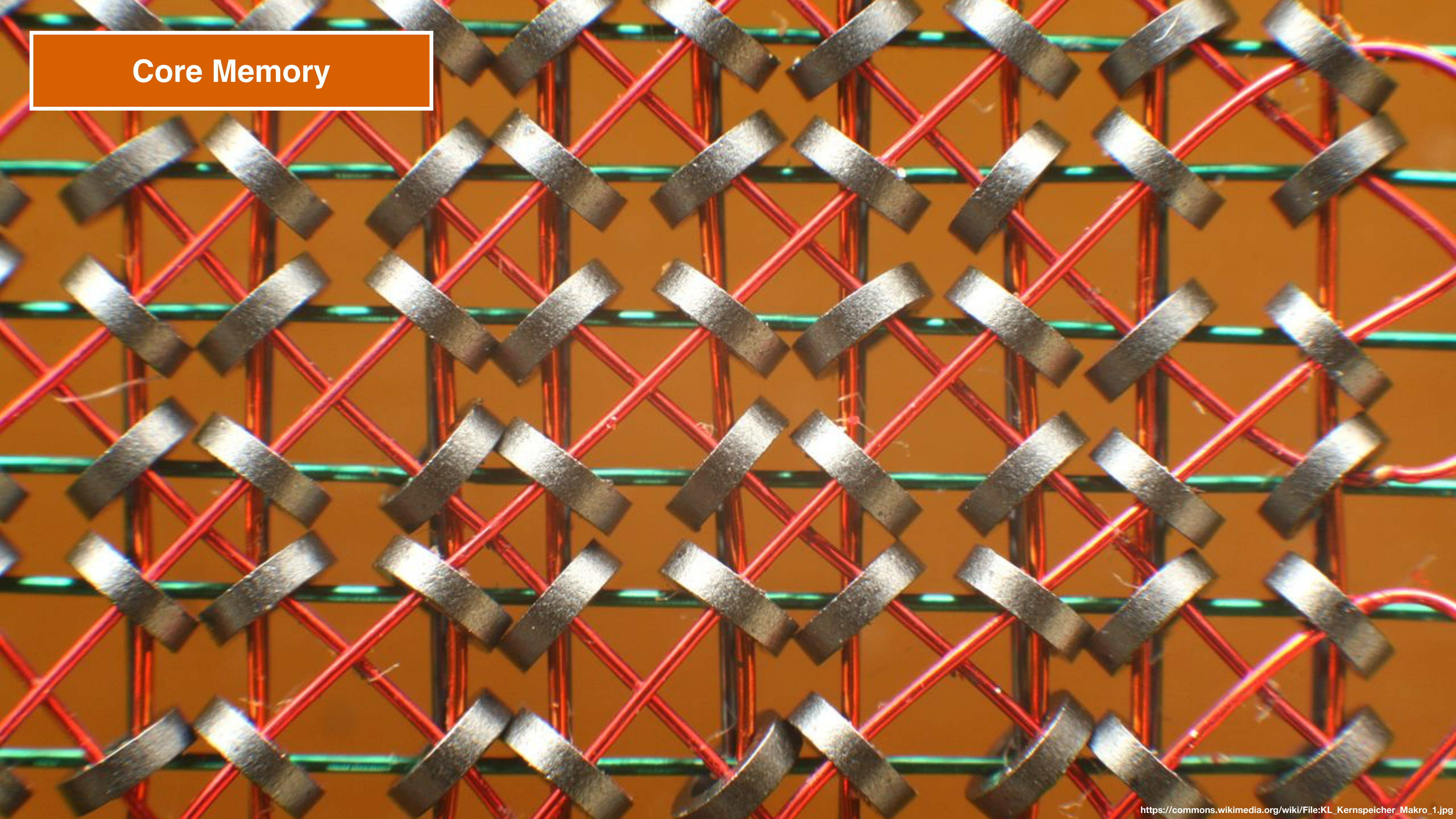
ROPE DRIVER MODULE B16-17

ROM

ROM

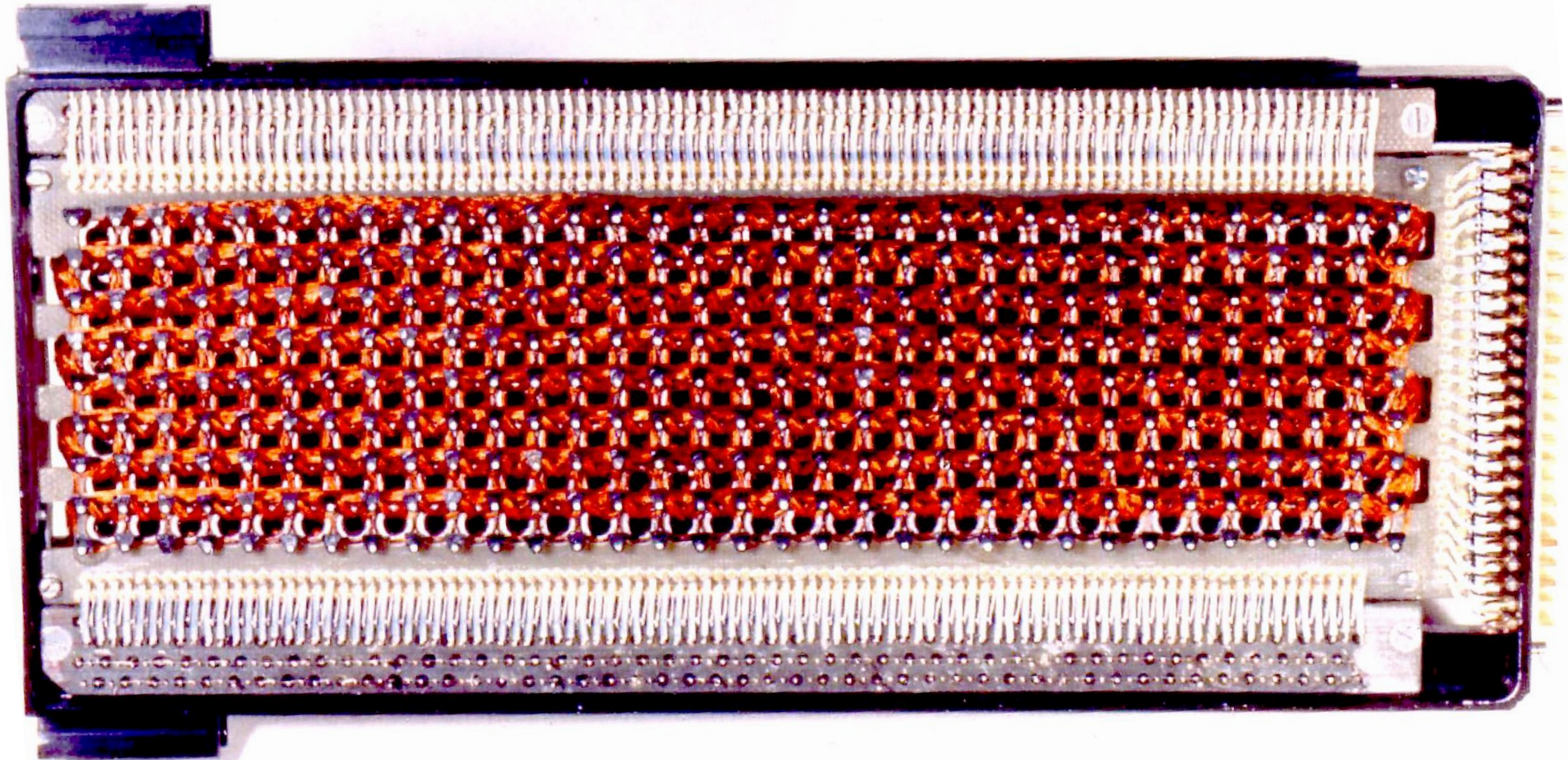


# Core Memory



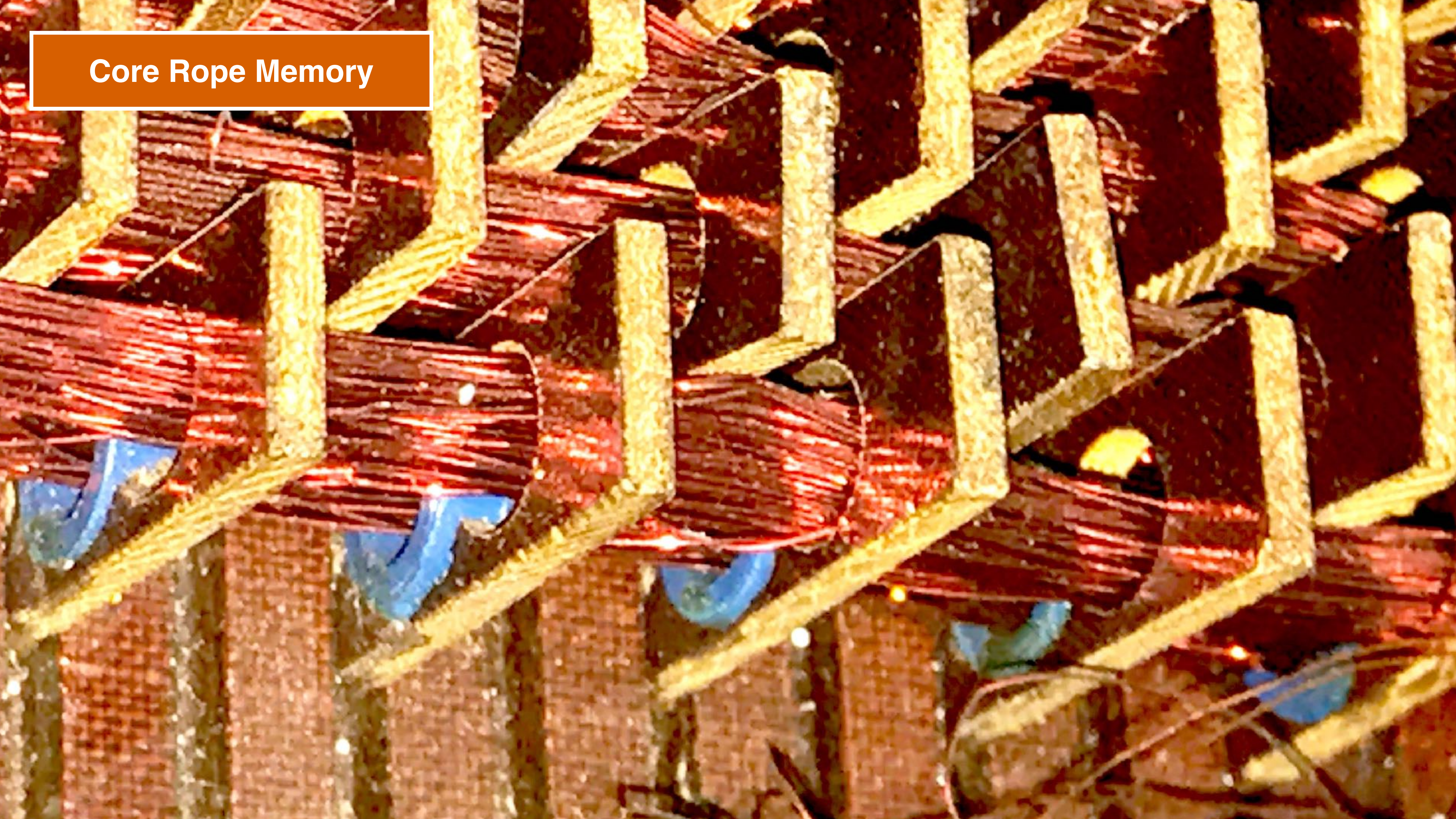


## Core Rope Memory





## Core Rope Memory



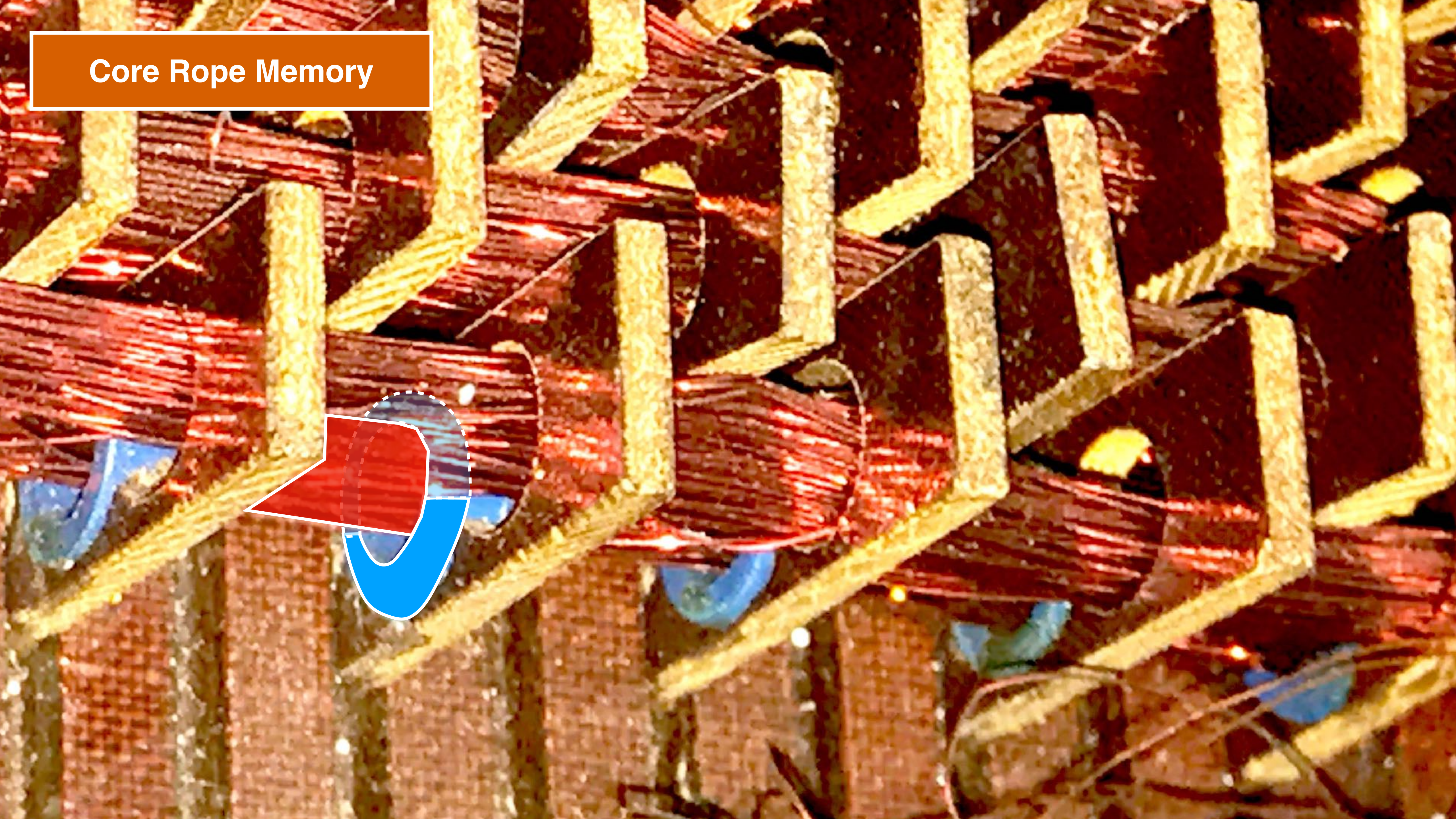


## Core Rope Memory



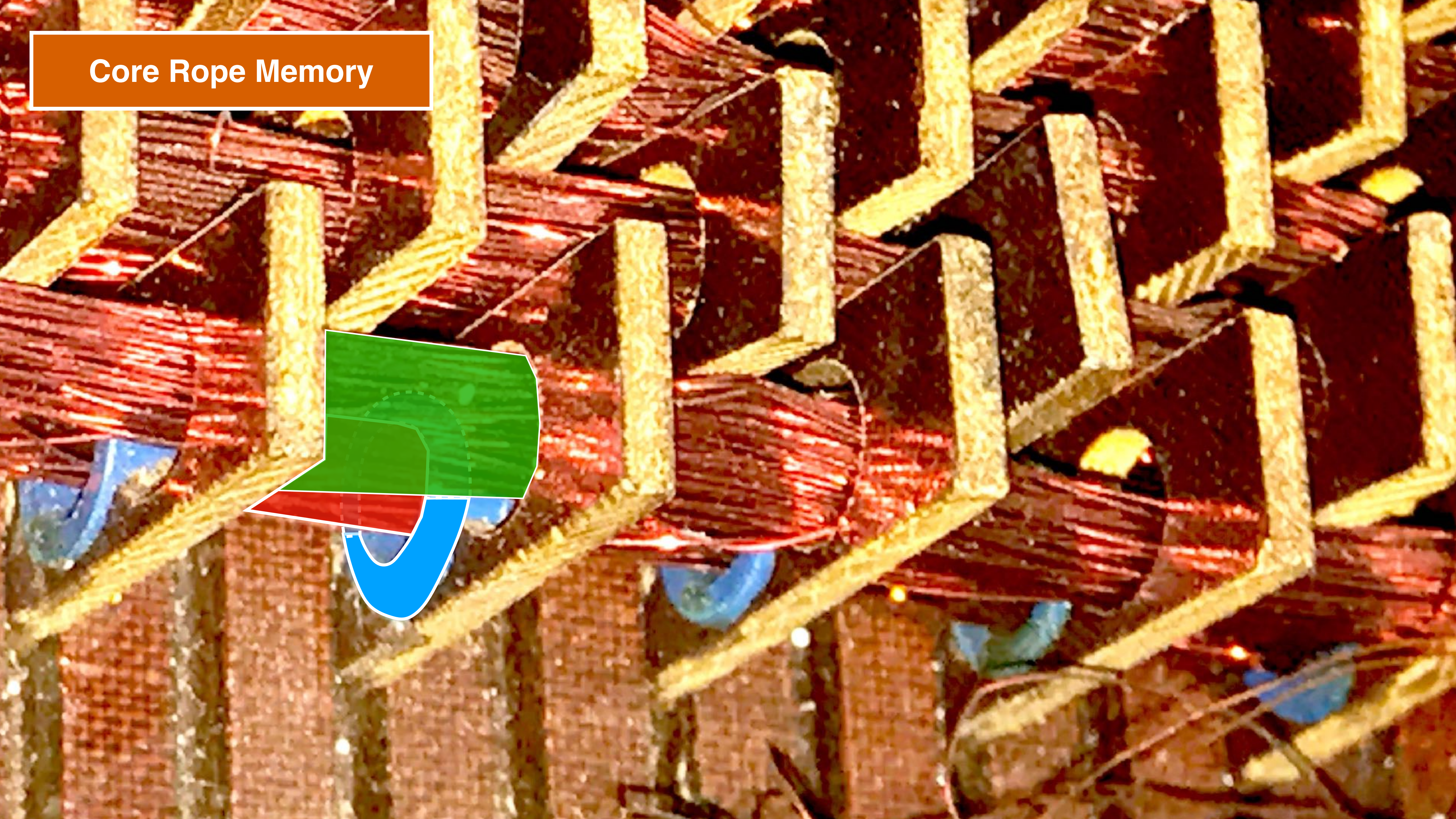


## Core Rope Memory





# Core Rope Memory



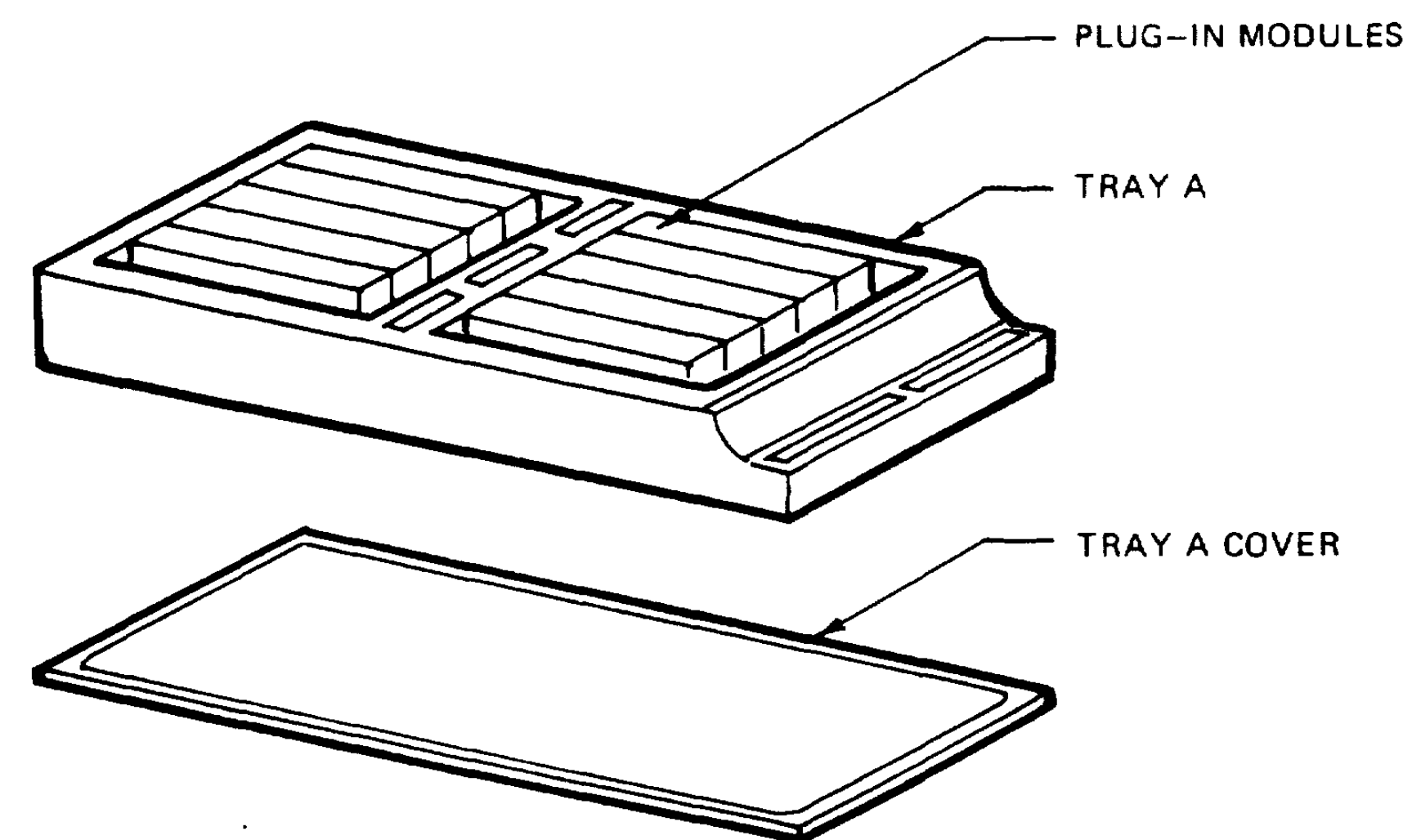


## Core Rope Memory

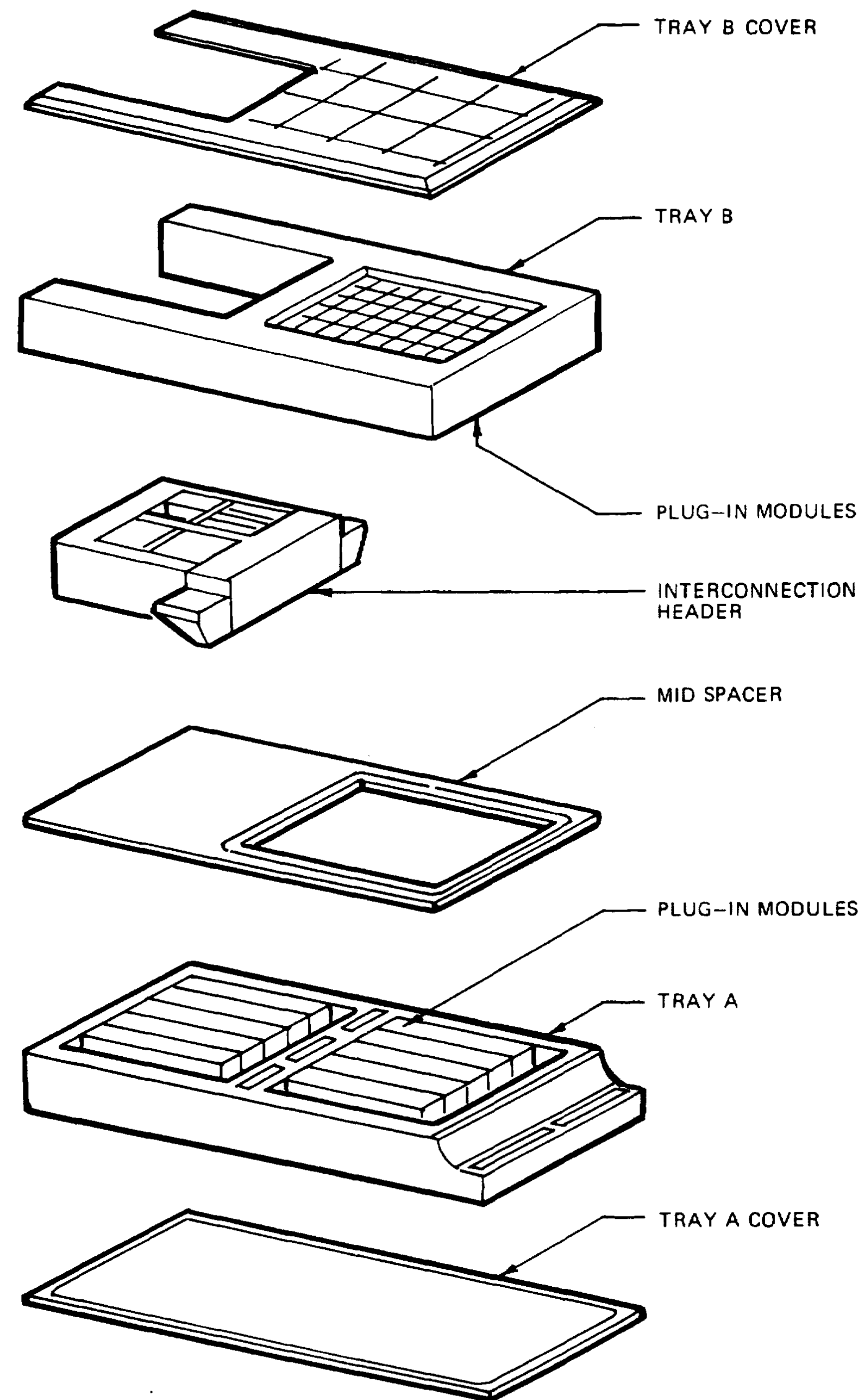




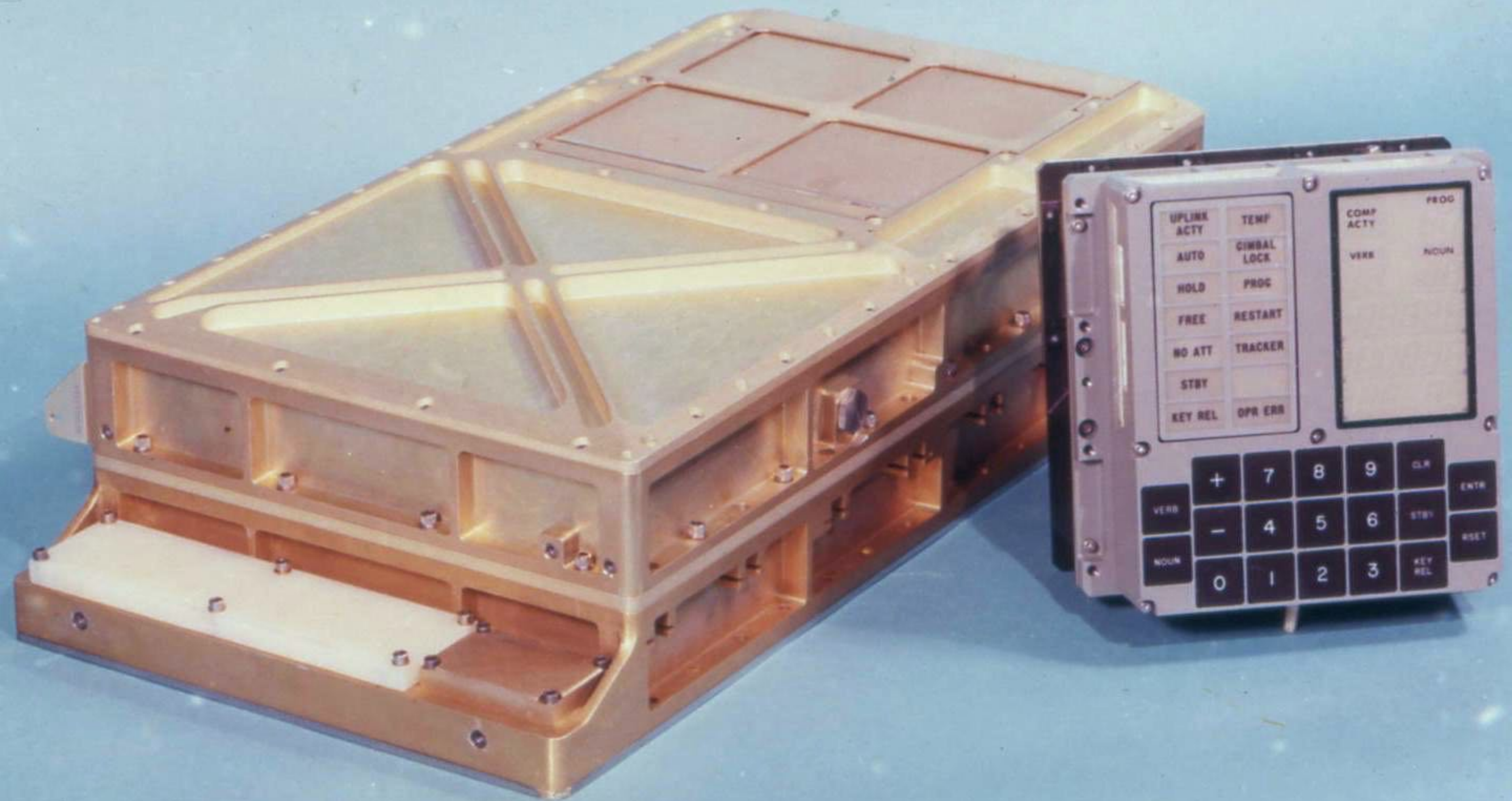




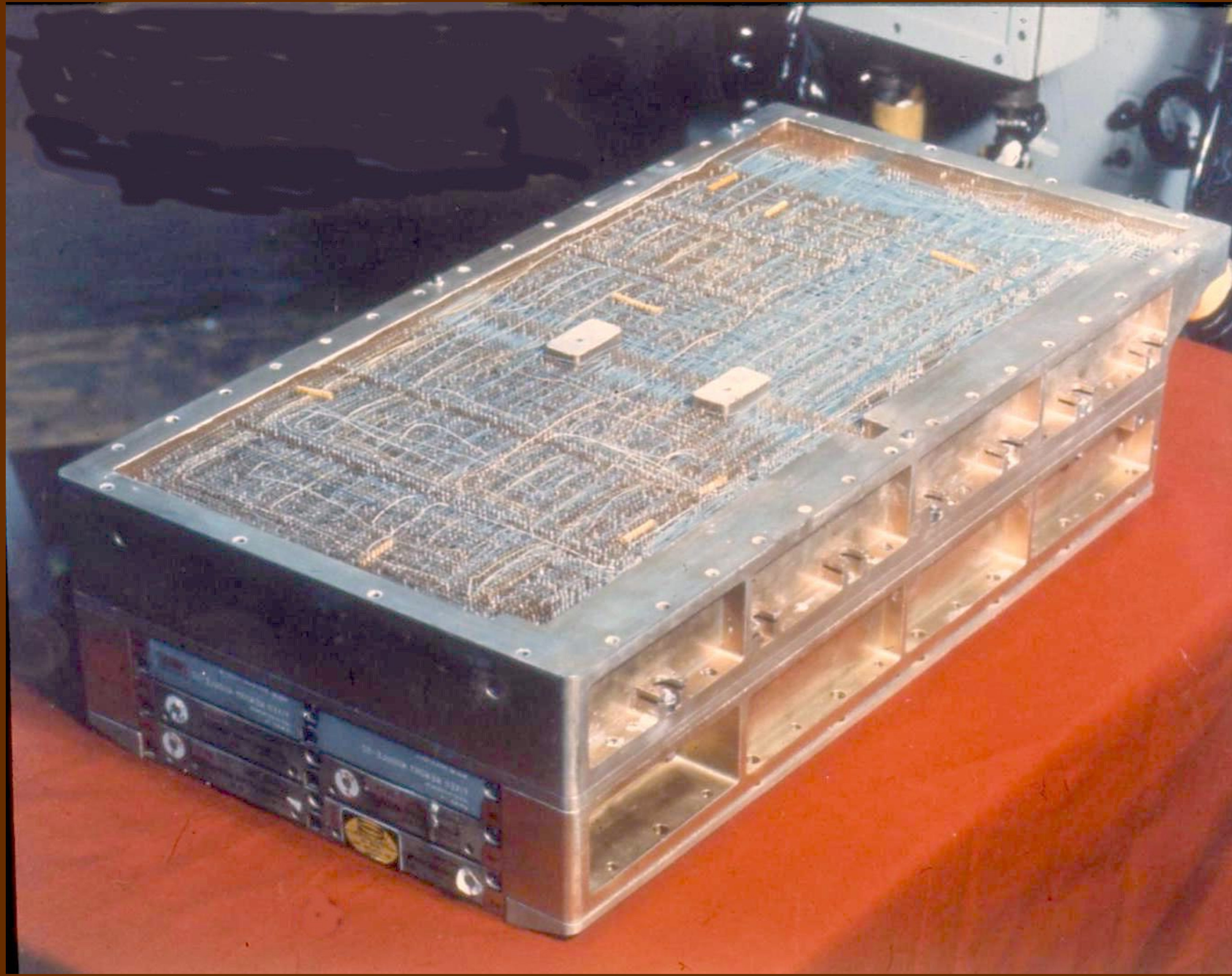




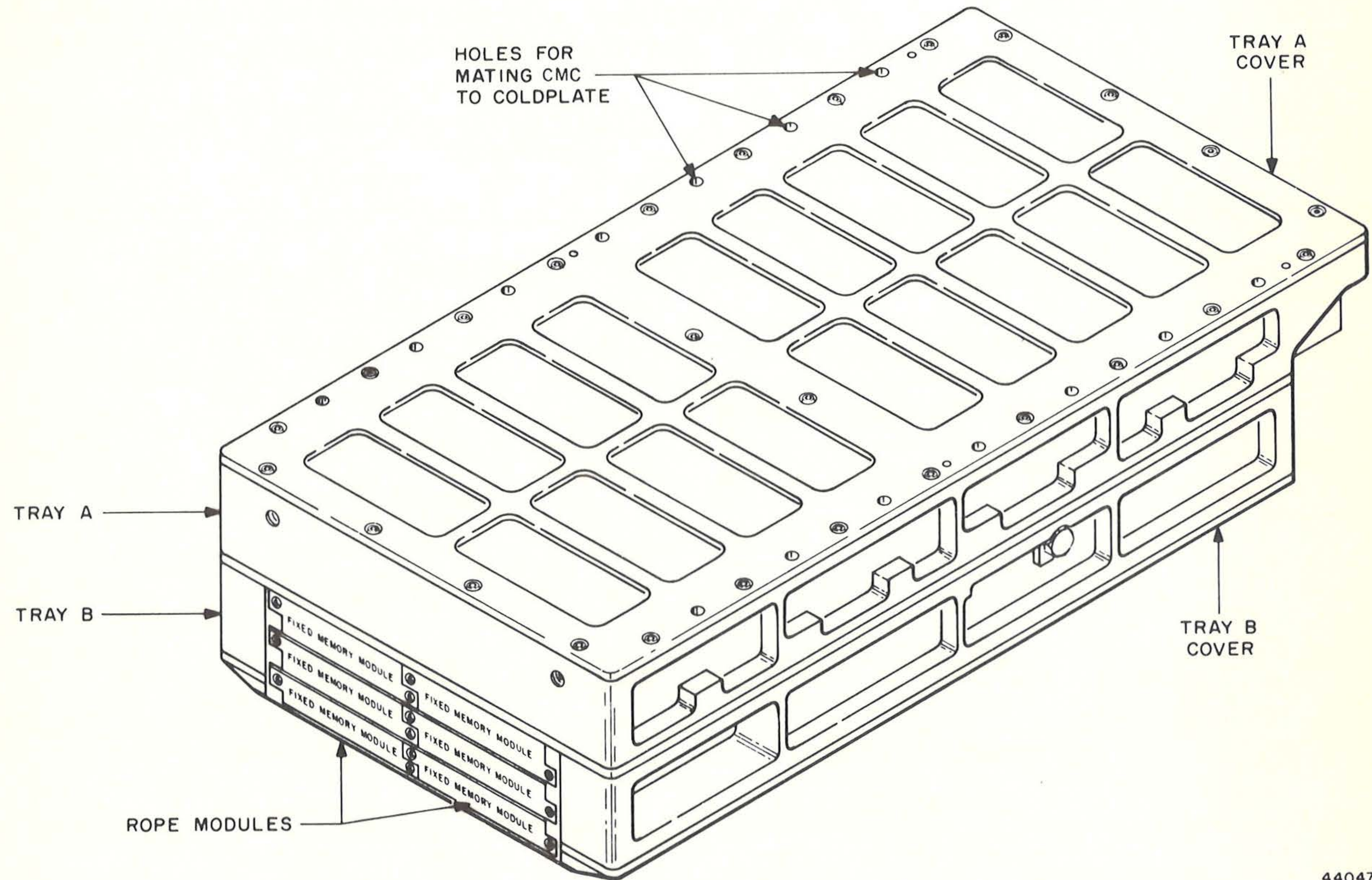










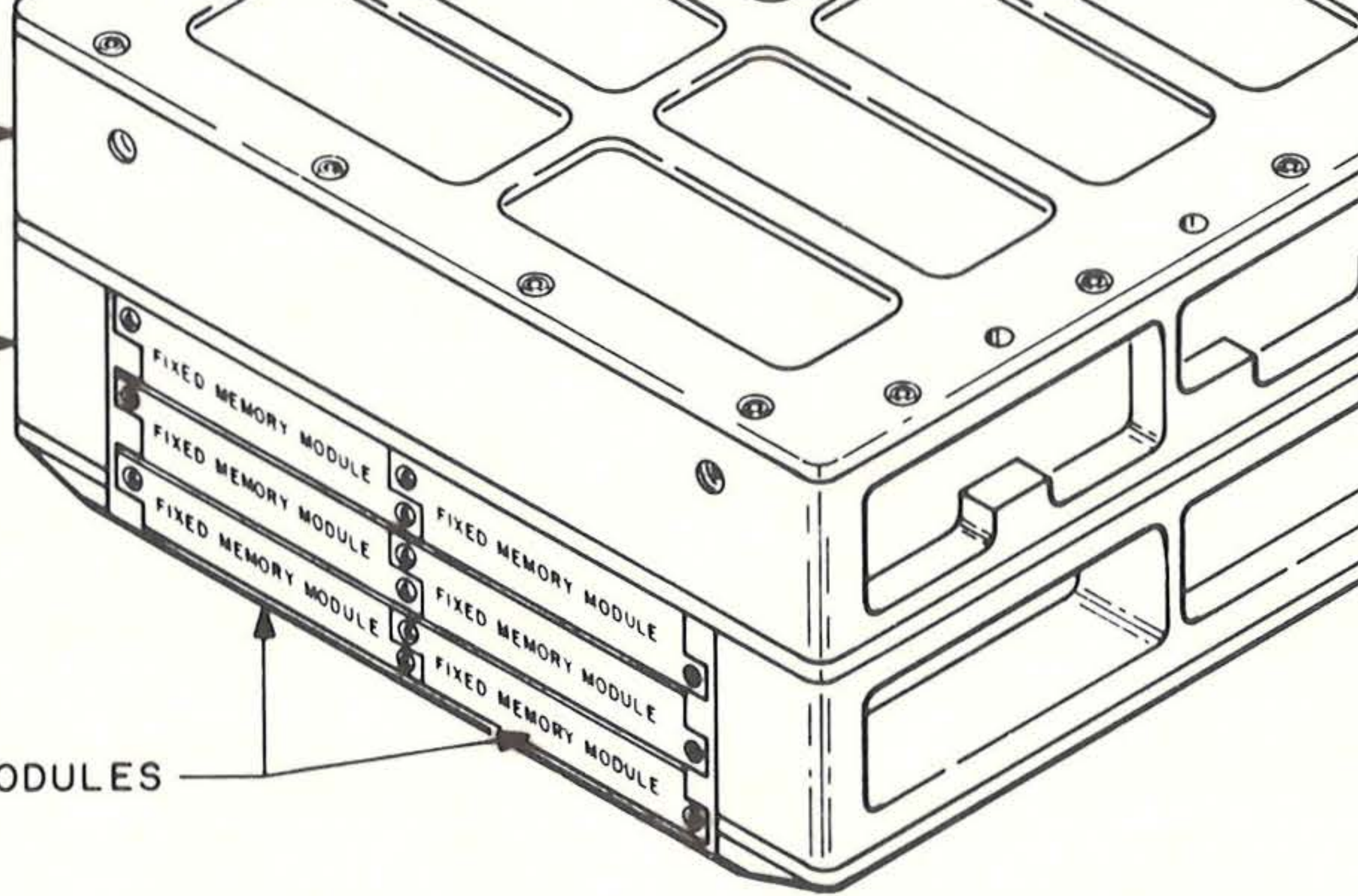




TRAY A

TRAY B

ROPE MODULES





TRAY A

TRAY B

ROPE MODULES

ROM

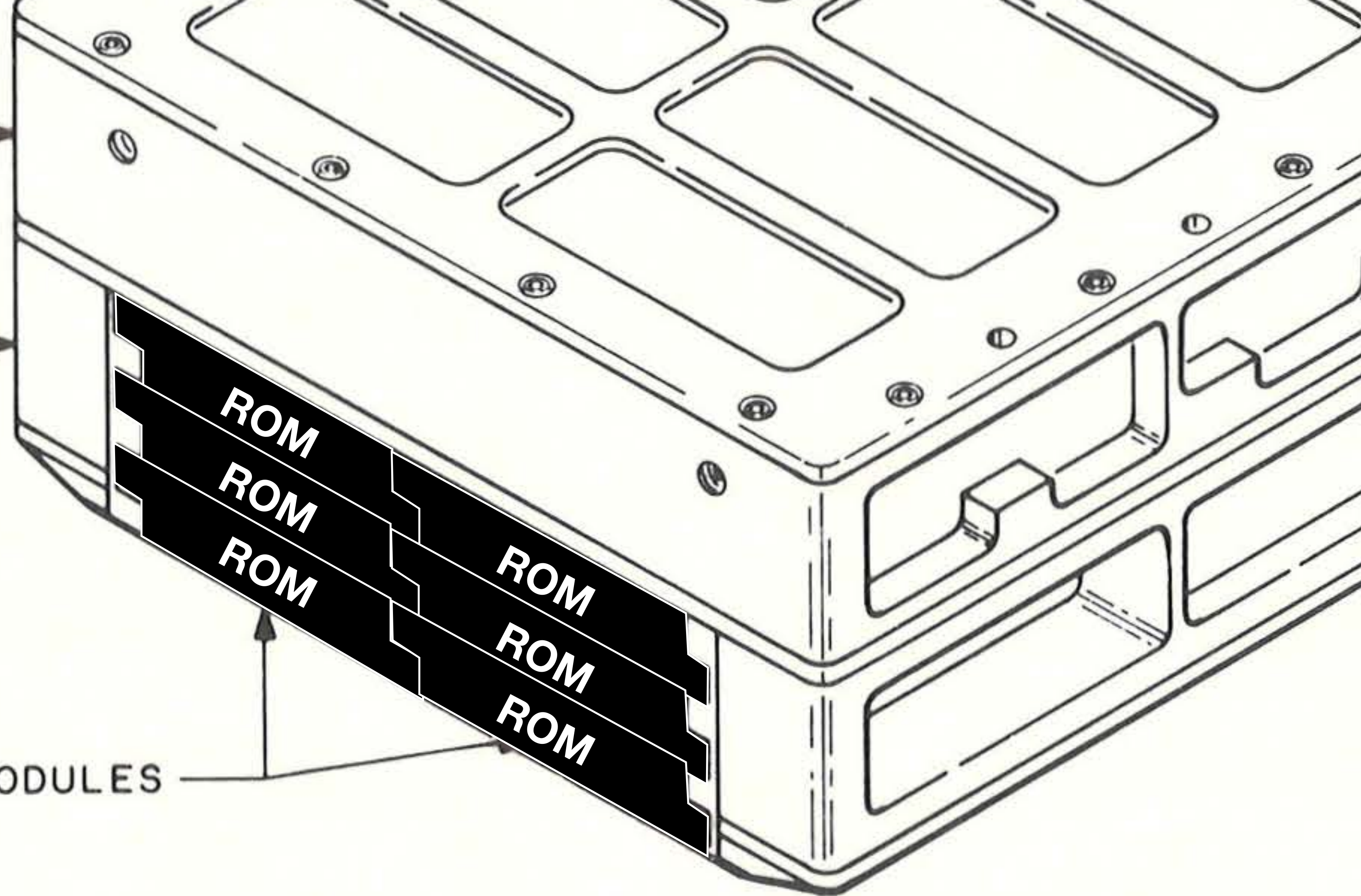
ROM

ROM

ROM

ROM

ROM



**Hardware**

**Timing**

**Schematics**

**Sequencer**

**ICs**

**Microcode**

**Modules**

**Unprogrammed Sequences**

**Trays**

**Hardware**



# Apollo Guidance Computer

Architecture

Mission Software

Hardware

Peripherals

System Software

# Peripherals

# Peripherals



**Peripherals**

**Gyroscope**

**Accelerometer**

**Optics**



## Peripherals

**Gyroscope**

**Landing Radar**

**Accelerometer**

**Rendezvous Radar**

**Optics**

**RCS Jets**

## Peripherals

Gyroscope

Landing Radar

DSKY

Accelerometer

Rendezvous Radar

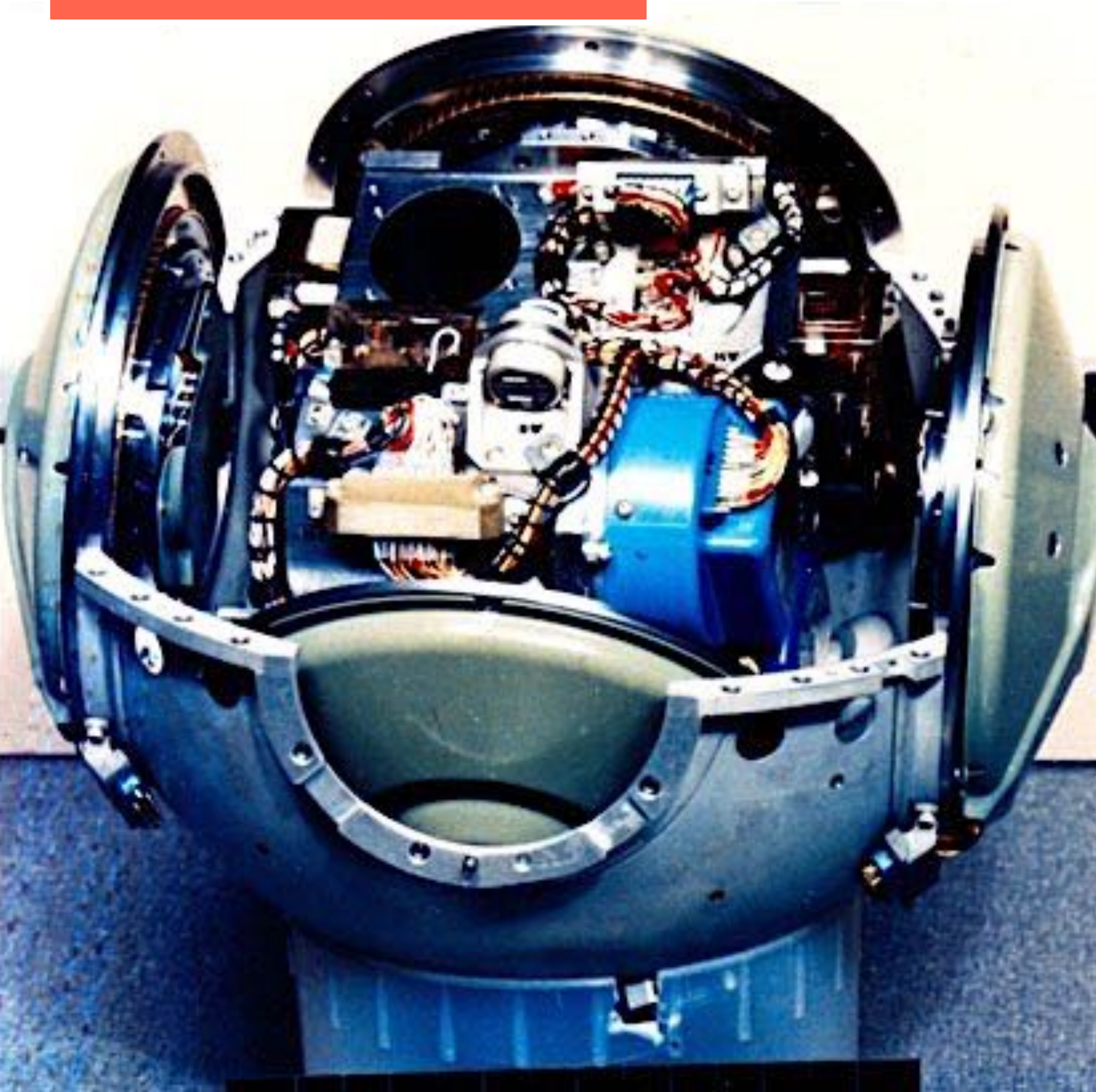
Uplink/Downlink

Optics

RCS Jets



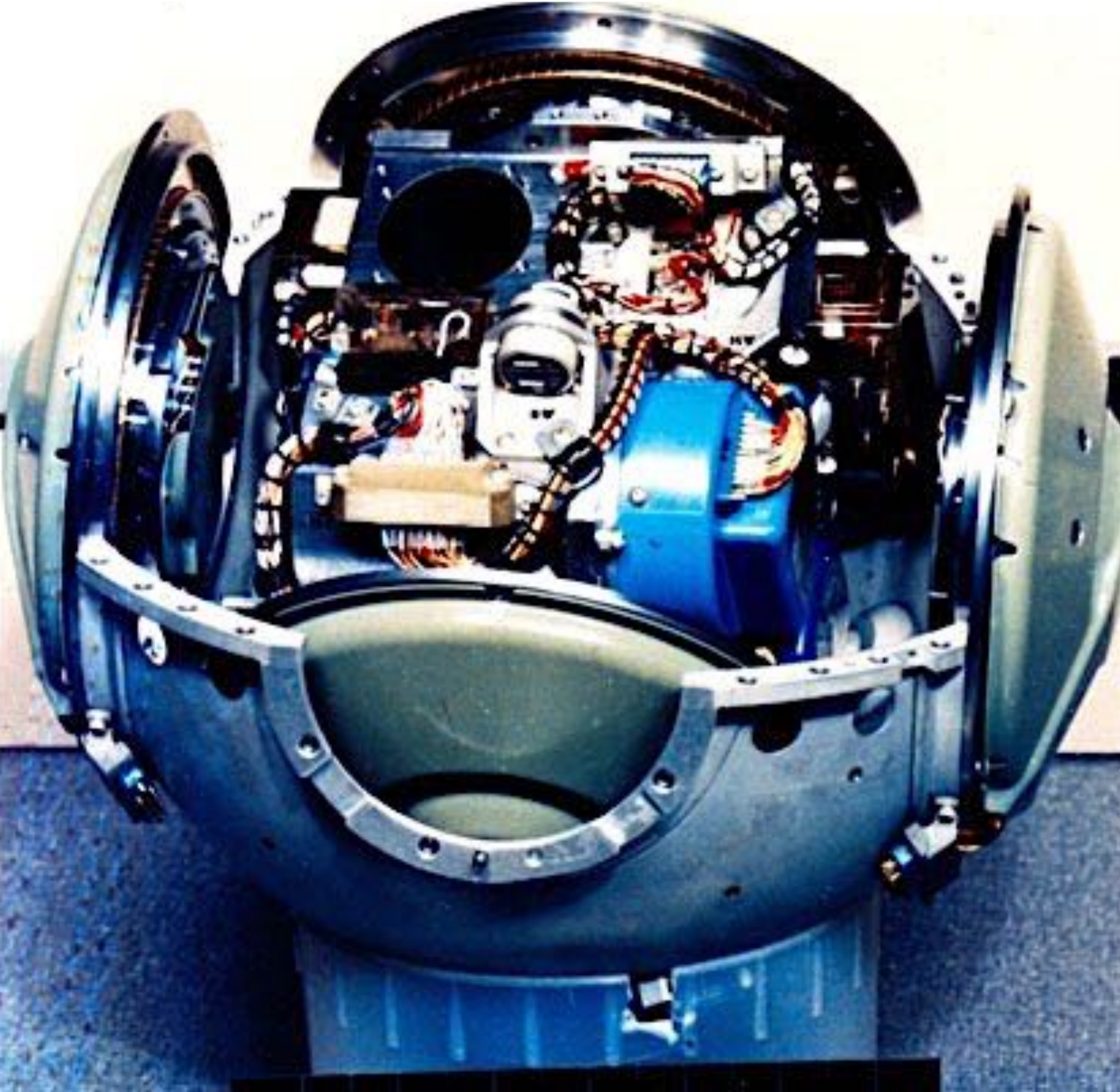
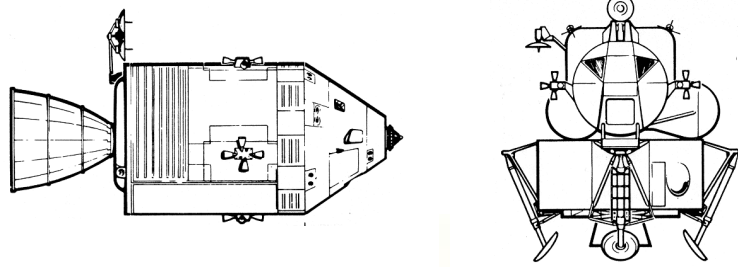
Gyroscope



Counters		
CDUX	01A	X gimbal
CDUY	01B	Y gimbal
CDUZ	01C	Z gimbal
CDUXCMD	028	X gimbal out
CDUYCMD	029	Y gimbal out
CDUZCMD	02A	Z gimbal out
I/O		
CHAN12	00A	3 coarse align enable 4 zero values 5 enable error ctr
CHAN14	00C	12-14 drive Z, Y, X
CHAN30	018	8 operate 10 cage 11 CDU fail 12 fail



# Gyroscope



Counters		
CDUX	01A	X gimbal
CDUY	01B	Y gimbal
CDUZ	01C	Z gimbal
CDUXCMD	028	X gimbal out
CDUYCMD	029	Y gimbal out
CDUZCMD	02A	Z gimbal out
I/O		
CHAN12	00A	3 coarse align enable 4 zero values 5 enable error ctr
CHAN14	00C	12-14 drive Z, Y, X
CHAN30	018	8 operate 10 cage 11 CDU fail 12 fail



Accelerometer



PIPAX

PIPAY

PIPAZ

Counters

	01D
	01E
	026

value X

value Y

value Z

I/O

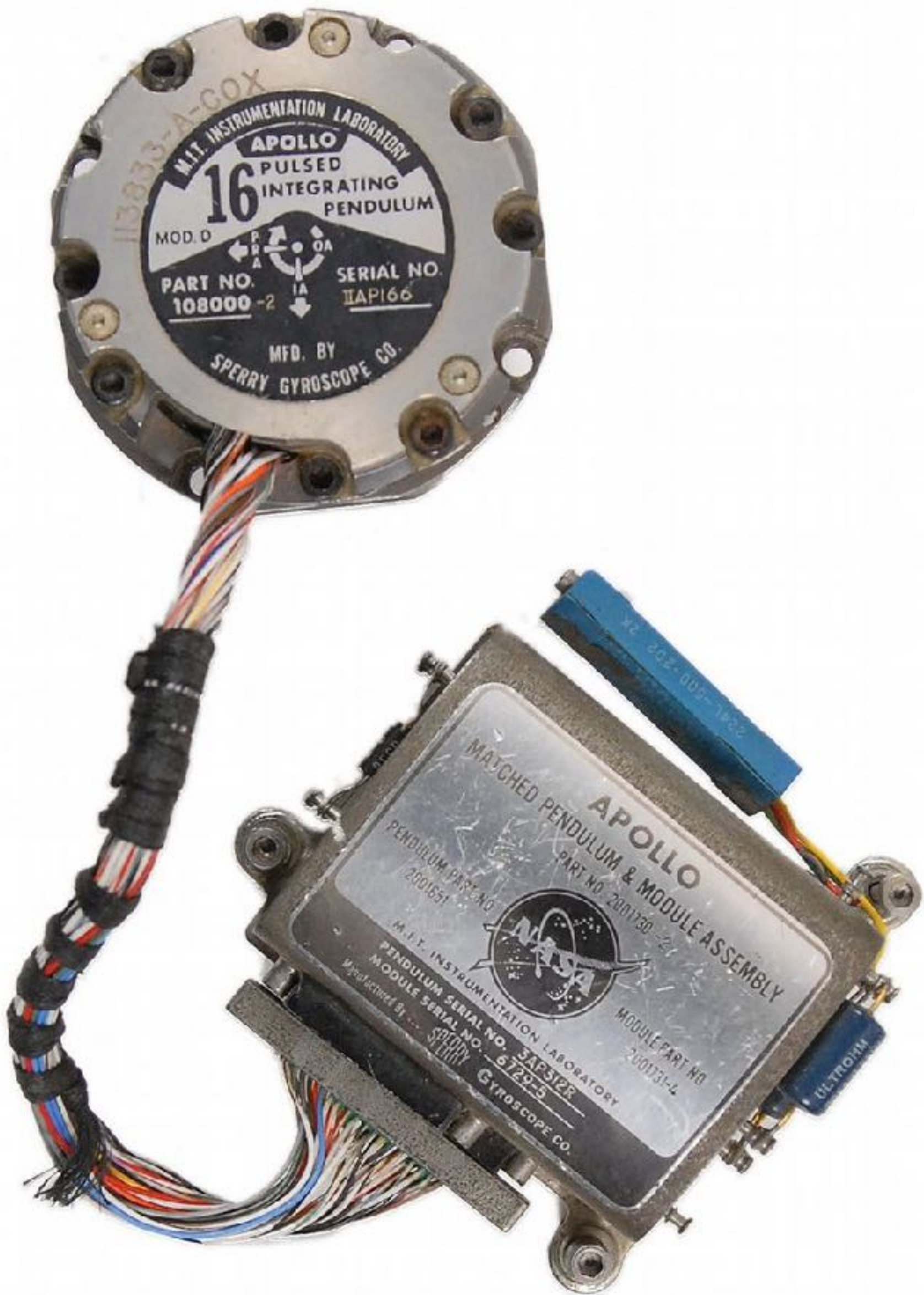
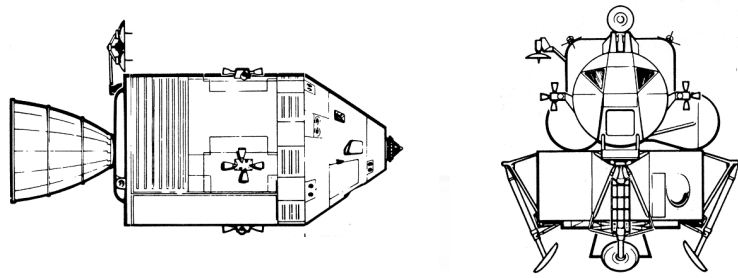
	01B
--	-----

12 fail

CHAN33



# Accelerometer



PIPAX

PIPAY

PIPAZ

CHAN33

Counters

	01D
	01E
	026

value X

value Y

value Z

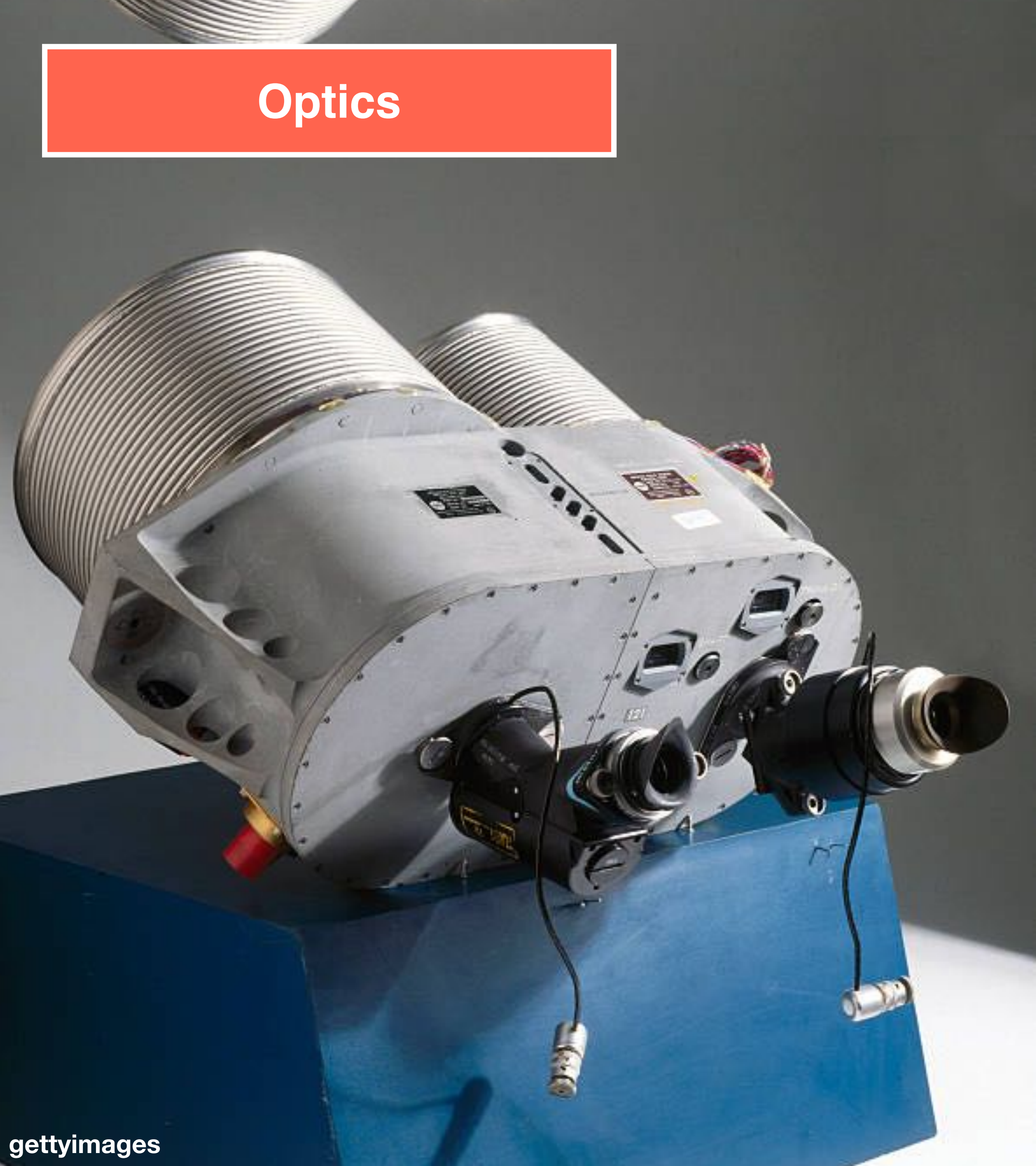
I/O

	01B
--	-----

12 fail



Optics



OPTY

OPTX

OPTYCMD

OPTXCMD

CHAN12

CHAN30

CHAN33

Counters

	01D
	01E
	02B
	02C

y axis in

x axis in

y axis out

x axis out

I/O

	00A
	018
	01B

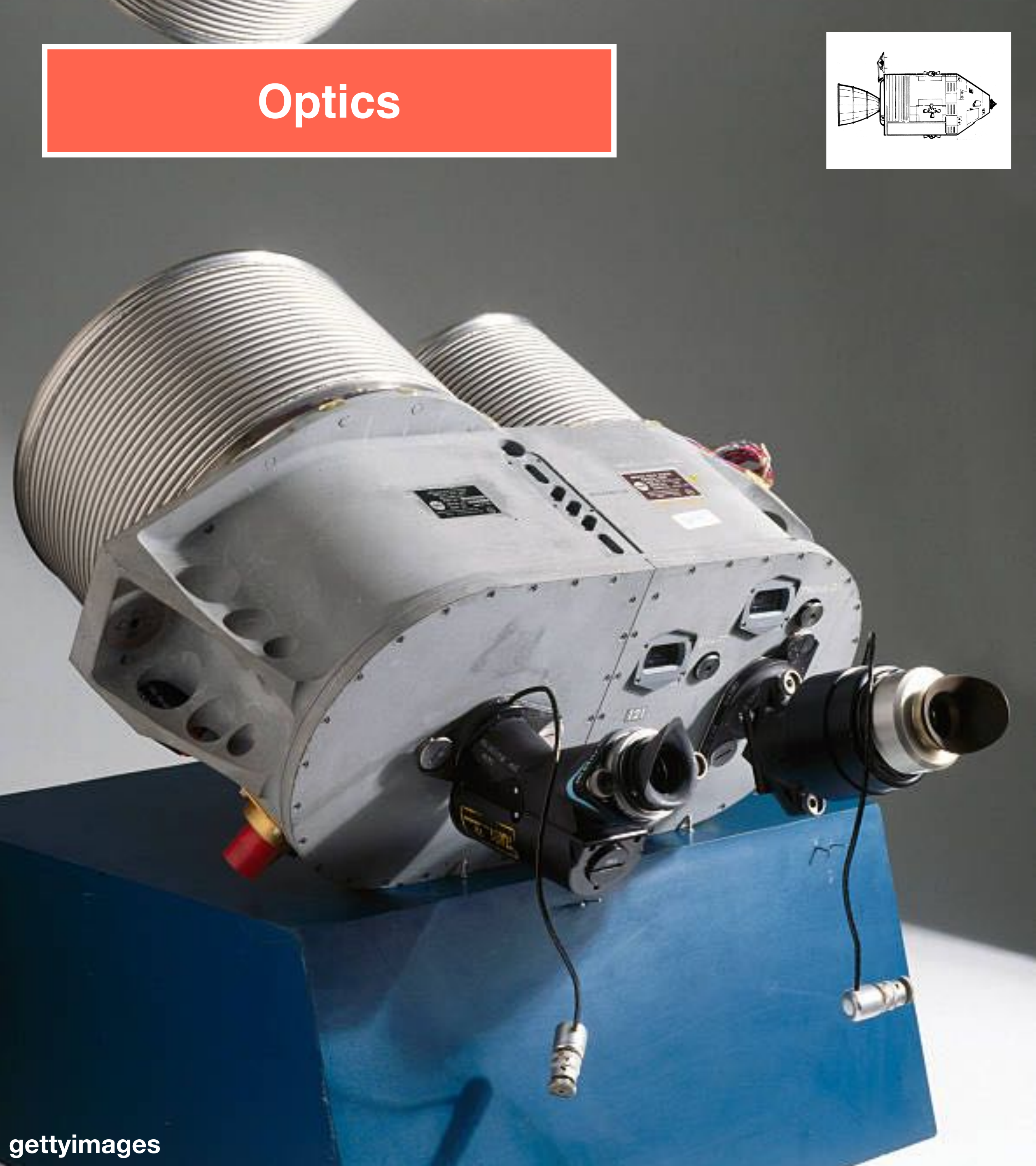
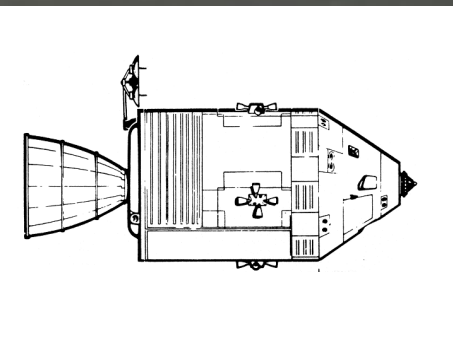
0 zero CDU  
1 enable error ctr  
9 zero optics  
10 disengage DAC

6 CDU fail

3 zero optics



Optics



OPTY  
OPTX  
OPTYCMD  
OPTXCMD

Counters

	01D
	01E
	02B
	02C

y axis in

x axis in

y axis out

x axis out

I/O

	00A
	018
	01B

0 zero CDU  
1 enable error ctr  
9 zero optics  
10 disengage DAC

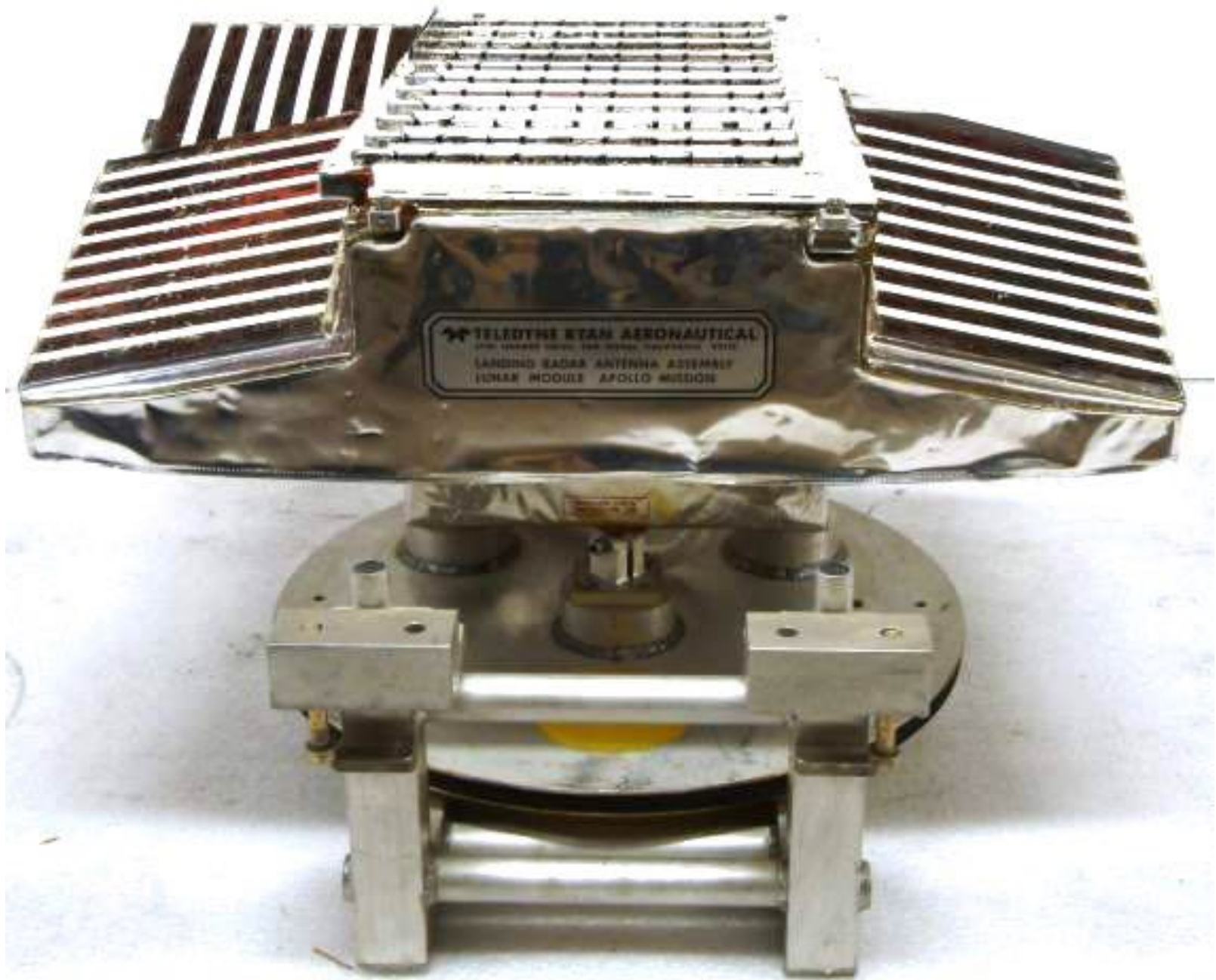
6 CDU fail

3 zero optics

CHAN12  
CHAN30  
CHAN33



# Landing Radar



**RADARUPT**

Interrupts

428

data ready

**RNRAD**

Counters

026

data

**CHAN13**

I/O

00B

0-2 radar select  
3 activity

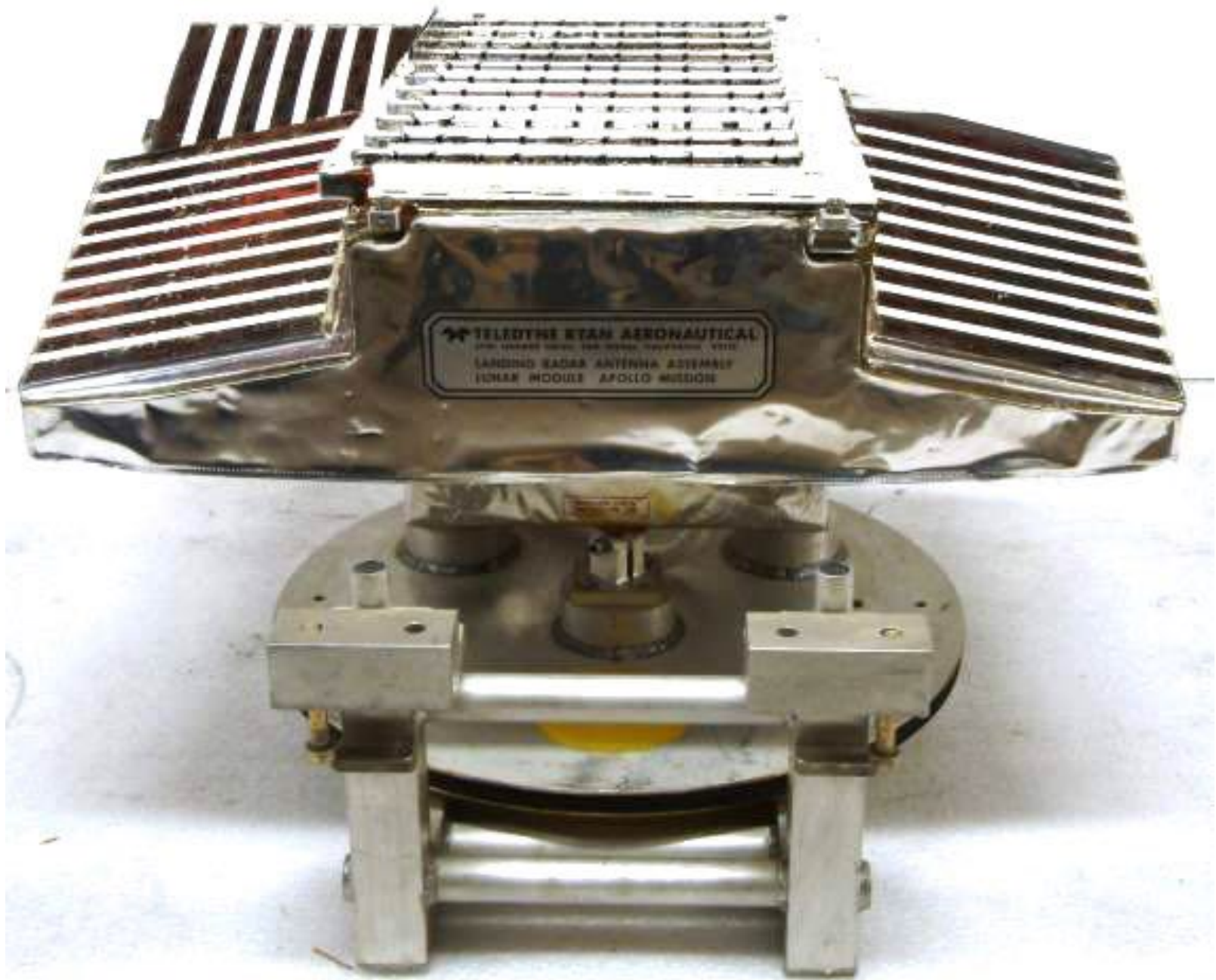
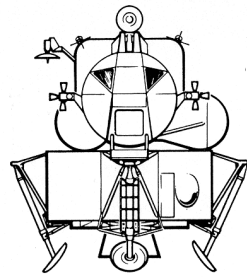
**CHAN33**

01B

1 auto-power on  
2 range low scale  
3 data good



# Landing Radar



**RADARUPT**

Interrupts

428

data ready

**RNRAD**

Counters

026

data

**CHAN13**

I/O

00B

0-2 radar select  
3 activity

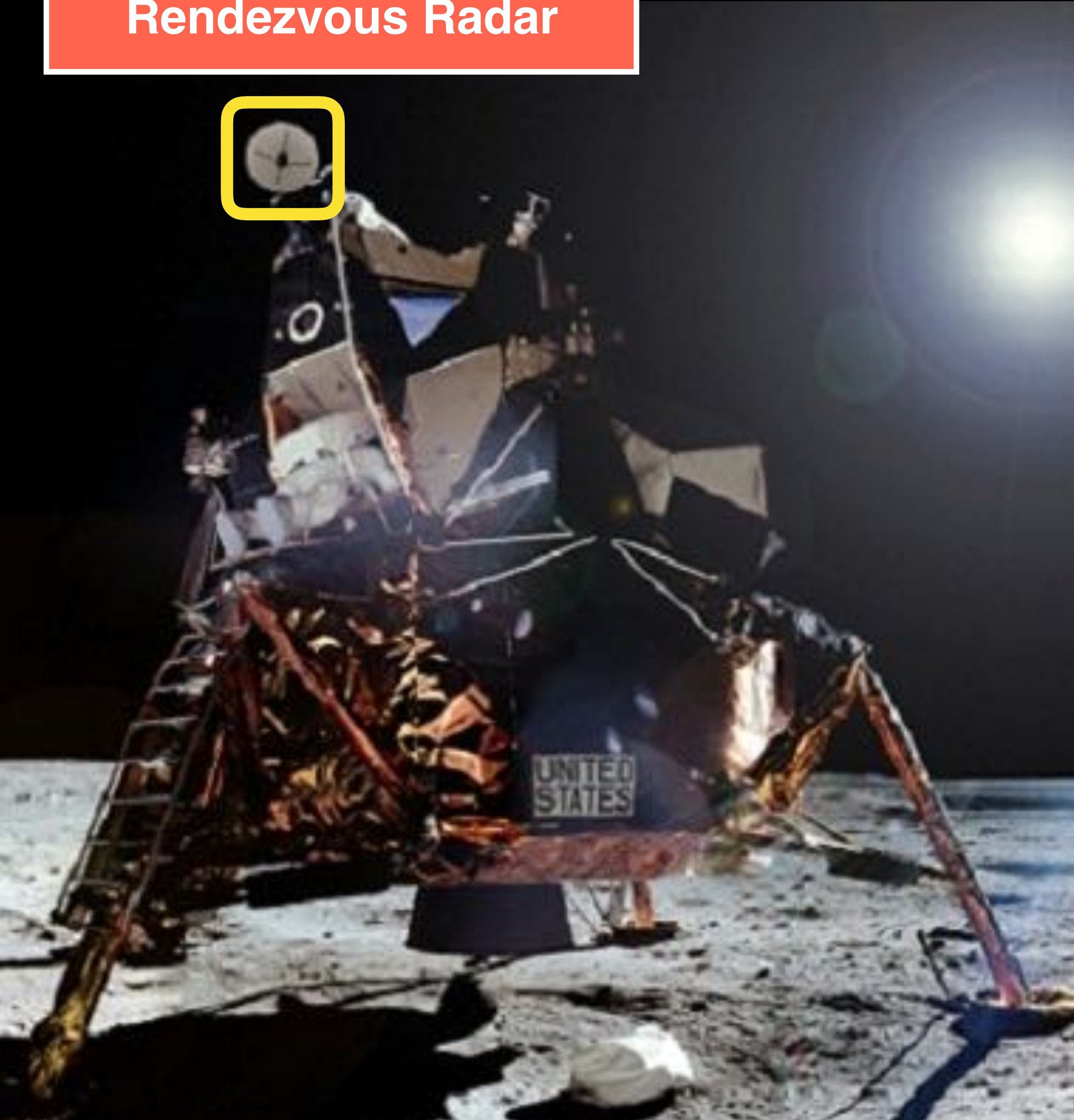
**CHAN33**

01B

1 auto-power on  
2 range low scale  
3 data good



Rendezvous Radar

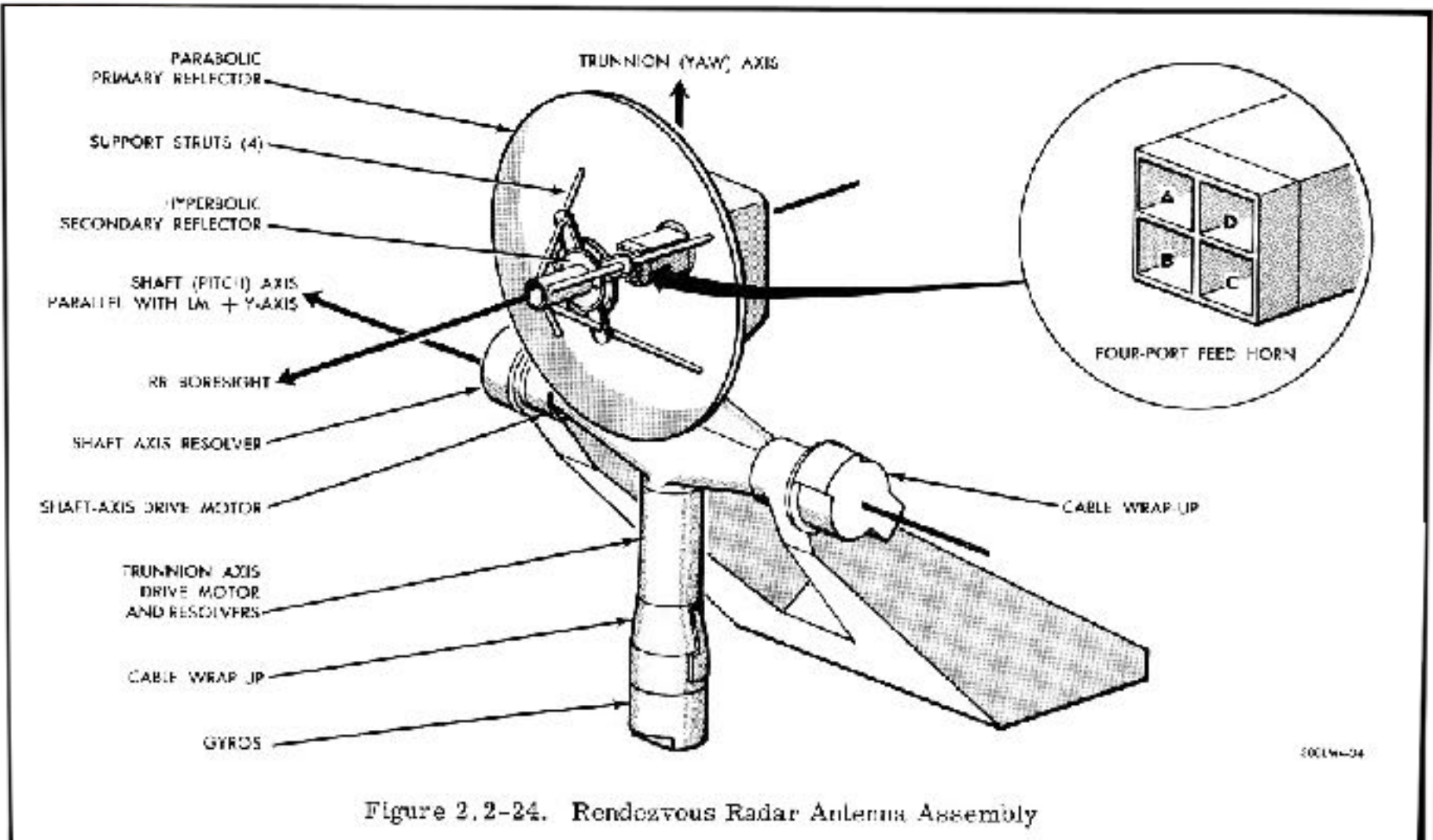
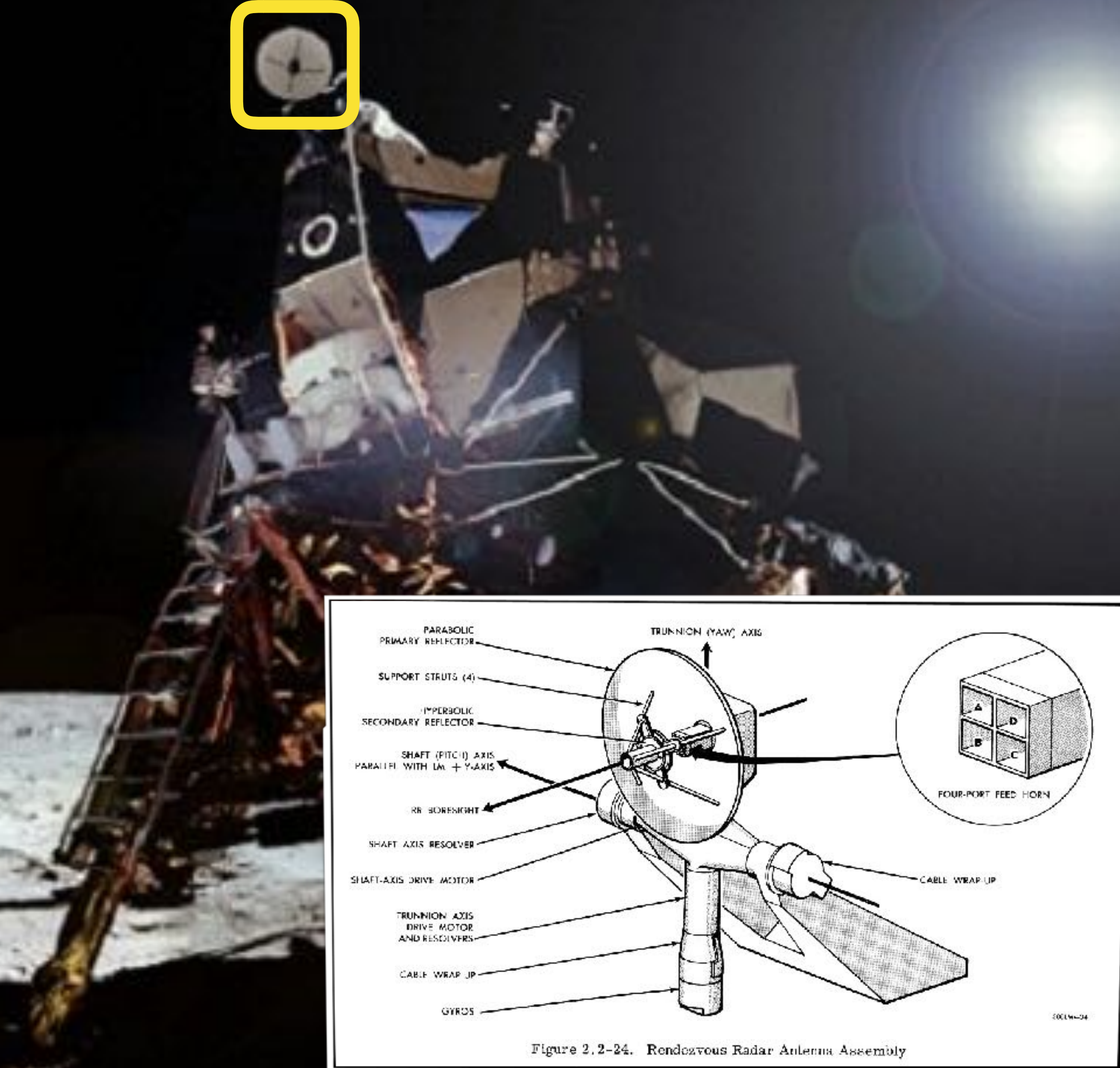
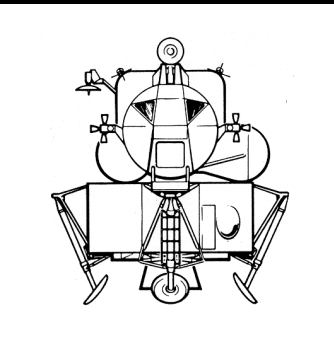


RADARUPT▶

Interrupts		
		428
		data ready
Counters		
CDUT		01D
		trunnion angle
CDUS		01E
		shaft angle
RNRAD		026
		data
I/O		
CHAN12		00A
		0 zero angles 1 enable error ctr 13 enbl auto track
CHAN13		00B
		0-2 radar select 3 activity
CHAN30		018
		6 CDU fail
CHAN33		01B
		1 auto-power on 2 range low scale 3 data good



# Rendezvous Radar



**RADARUPT**

Interrupts

	428
--	-----

data ready

Counters

**CDUT**  
**CDUS**  
**RNRAD**

	01D
	01E
	026

trunnion angle

shaft angle

data

I/O

**CHAN12**  
**CHAN13**  
**CHAN30**  
**CHAN33**

	00A
	00B
	018
	01B

0 zero angles  
1 enable error ctr  
13 enbl auto track

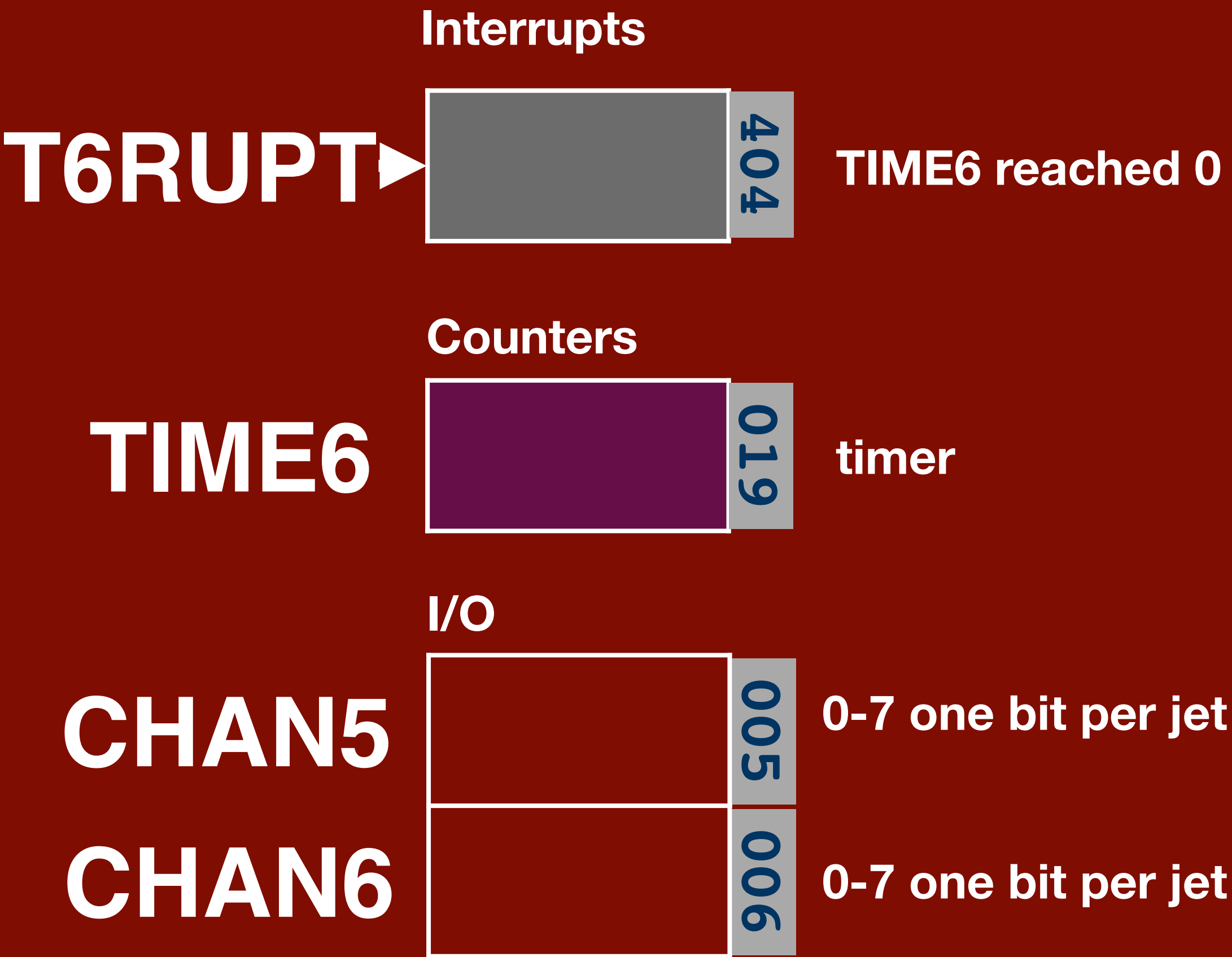
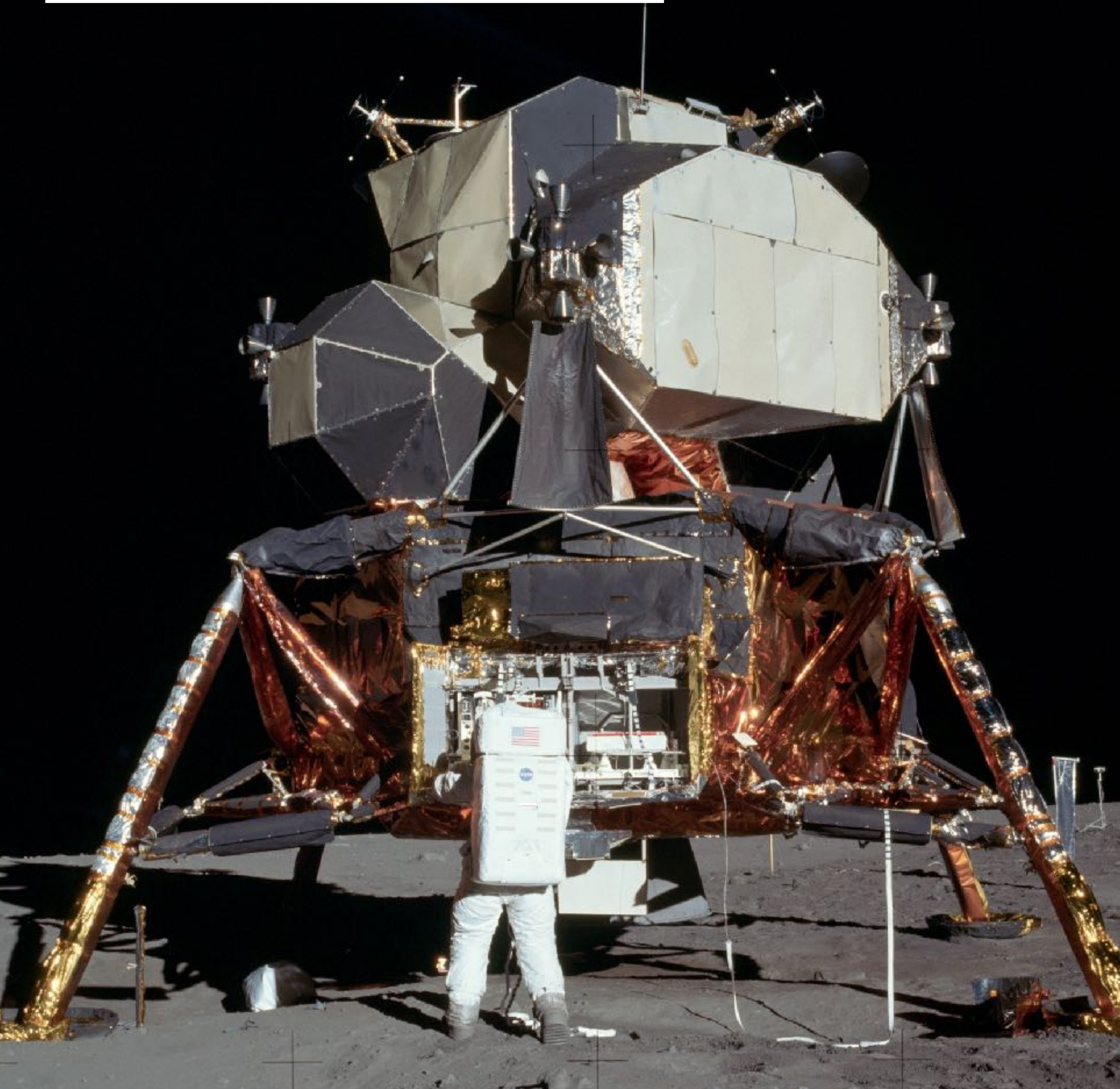
0-2 radar select  
3 activity

6 CDU fail

1 auto-power on  
2 range low scale  
3 data good

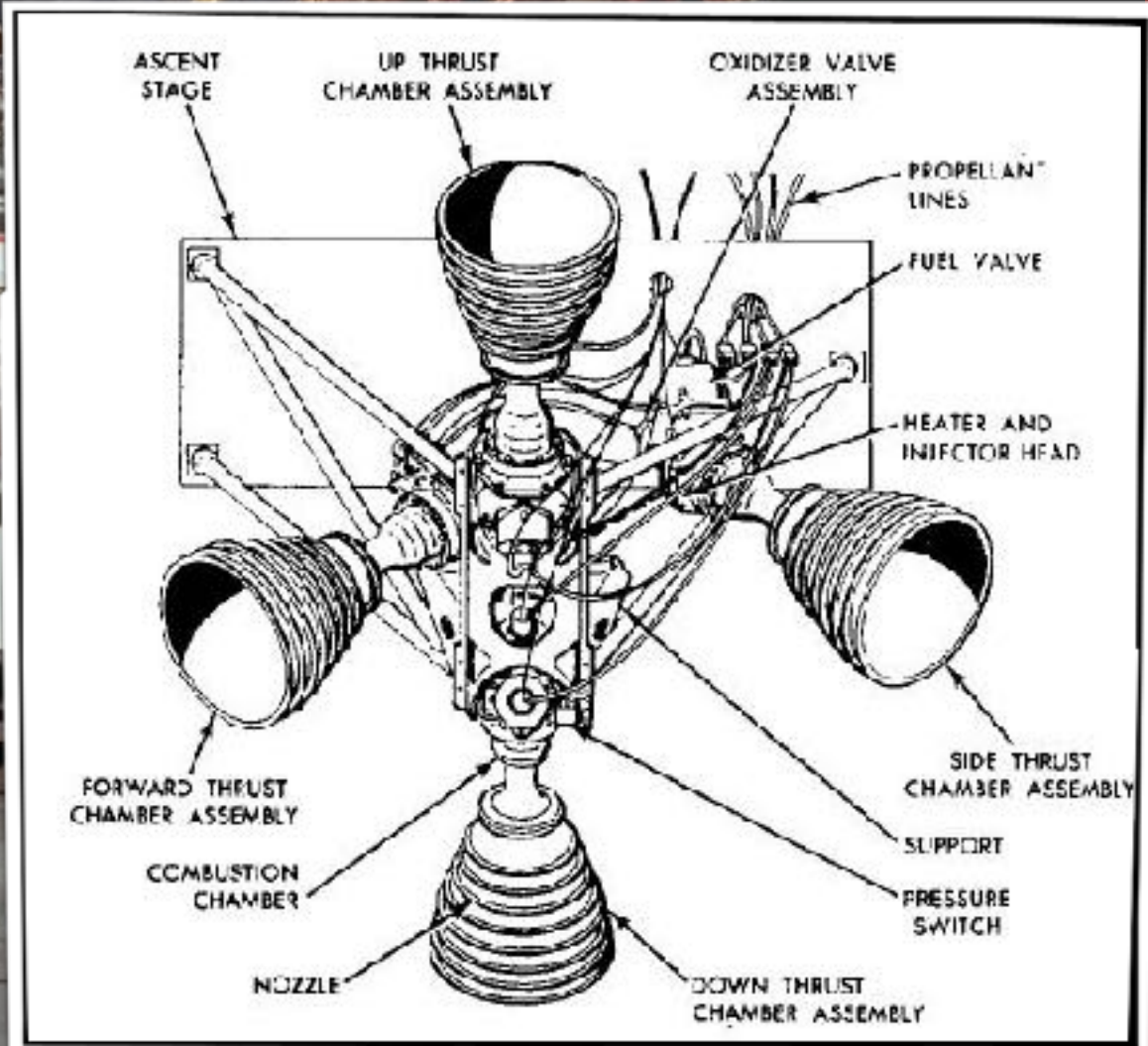
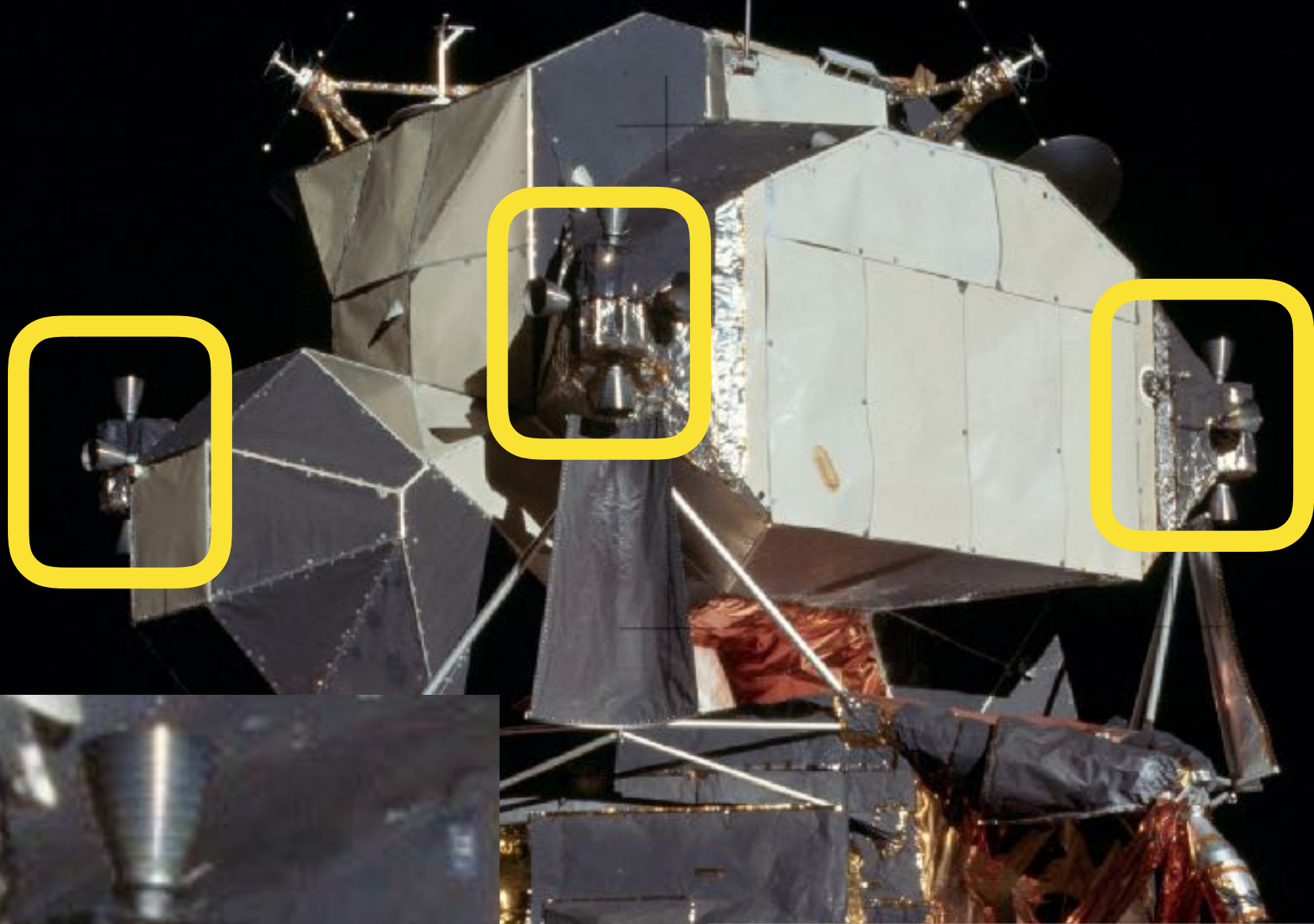
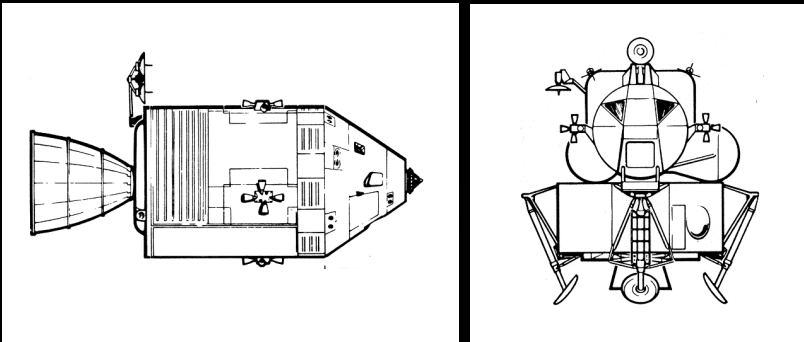


RCS Jets







RCS Jets




Interrupts


**T6RUPT**  **404** TIME6 reached 0

Counters

**TIME6**  **019** timer

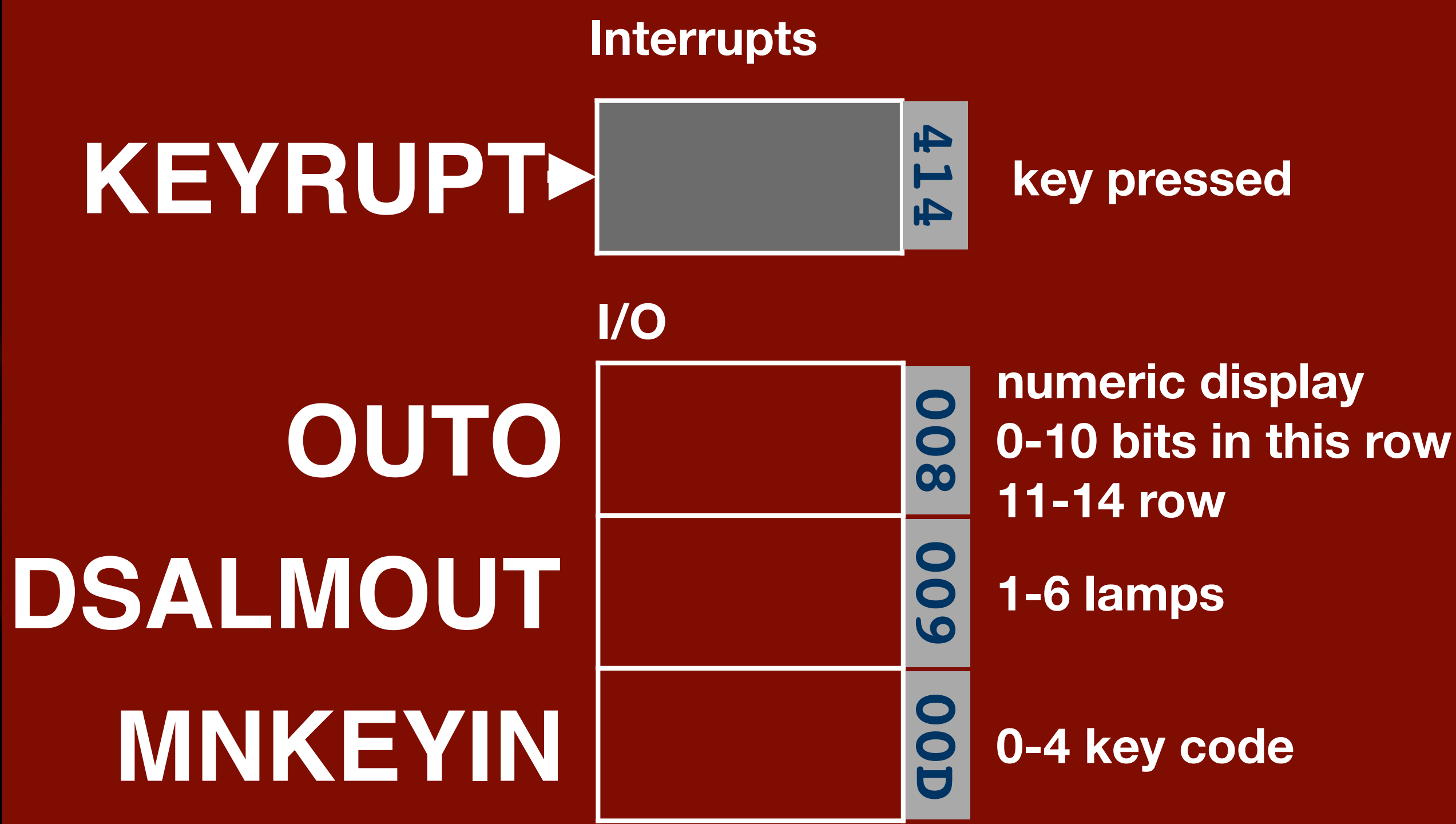
I/O

**CHAN5**  **005** 0-7 one bit per jet

**CHAN6**  **006** 0-7 one bit per jet



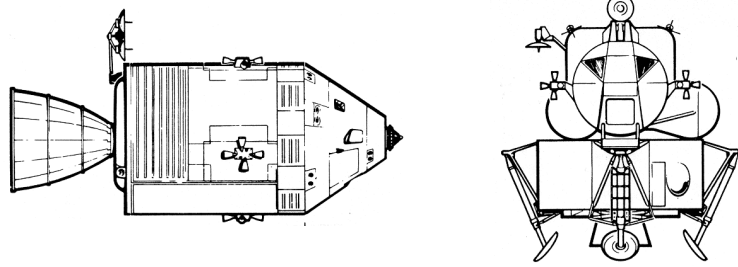
DSKY



The Command Module contains two DSKYs, so it has a second set of registers.



# DSKY



KEYRUPT

Interrupts

414

key pressed

OUTO

I/O

008

numeric display  
0-10 bits in this row  
11-14 row

DSALMOUT

009

1-6 lamps

MNKEYIN

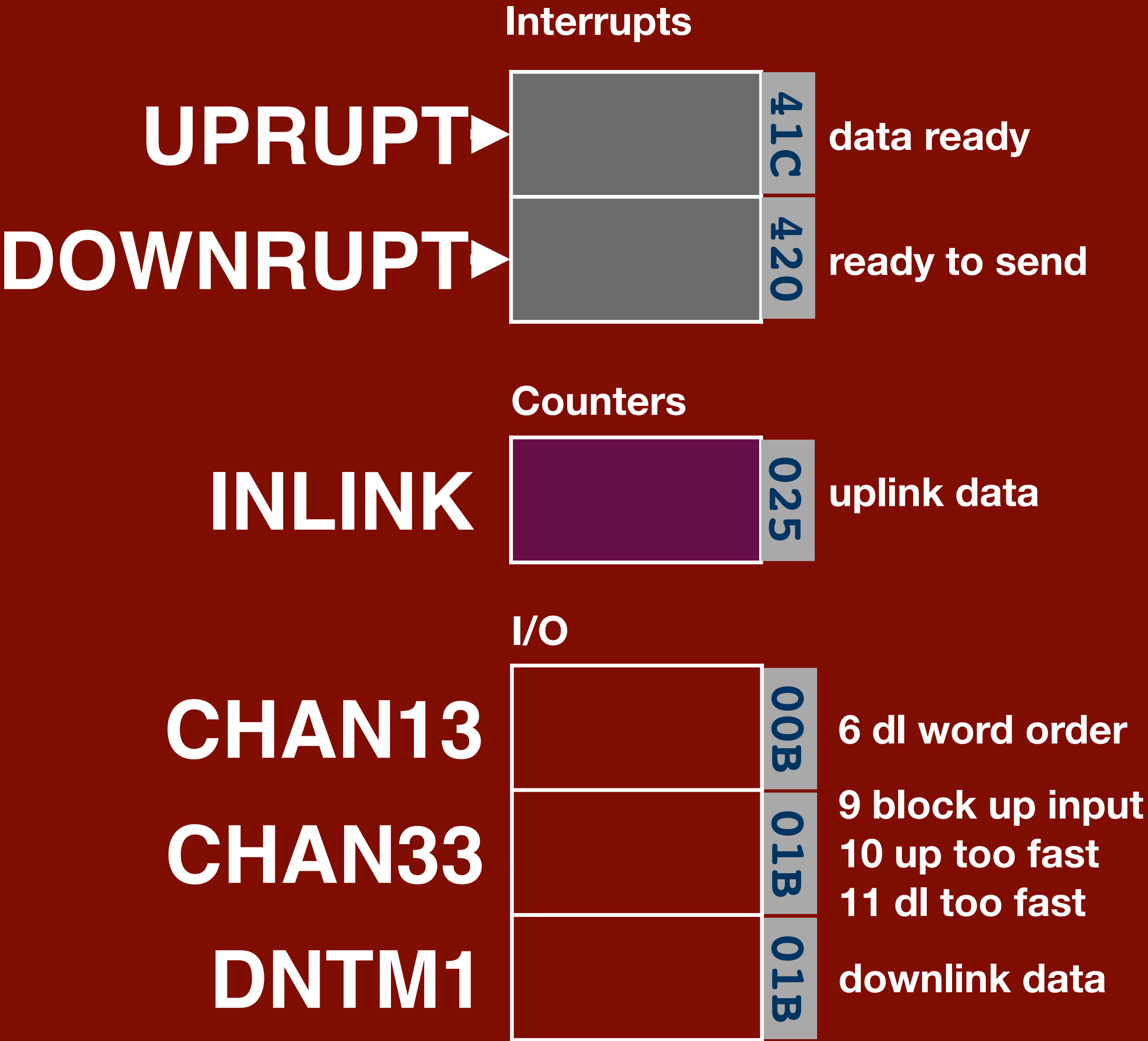
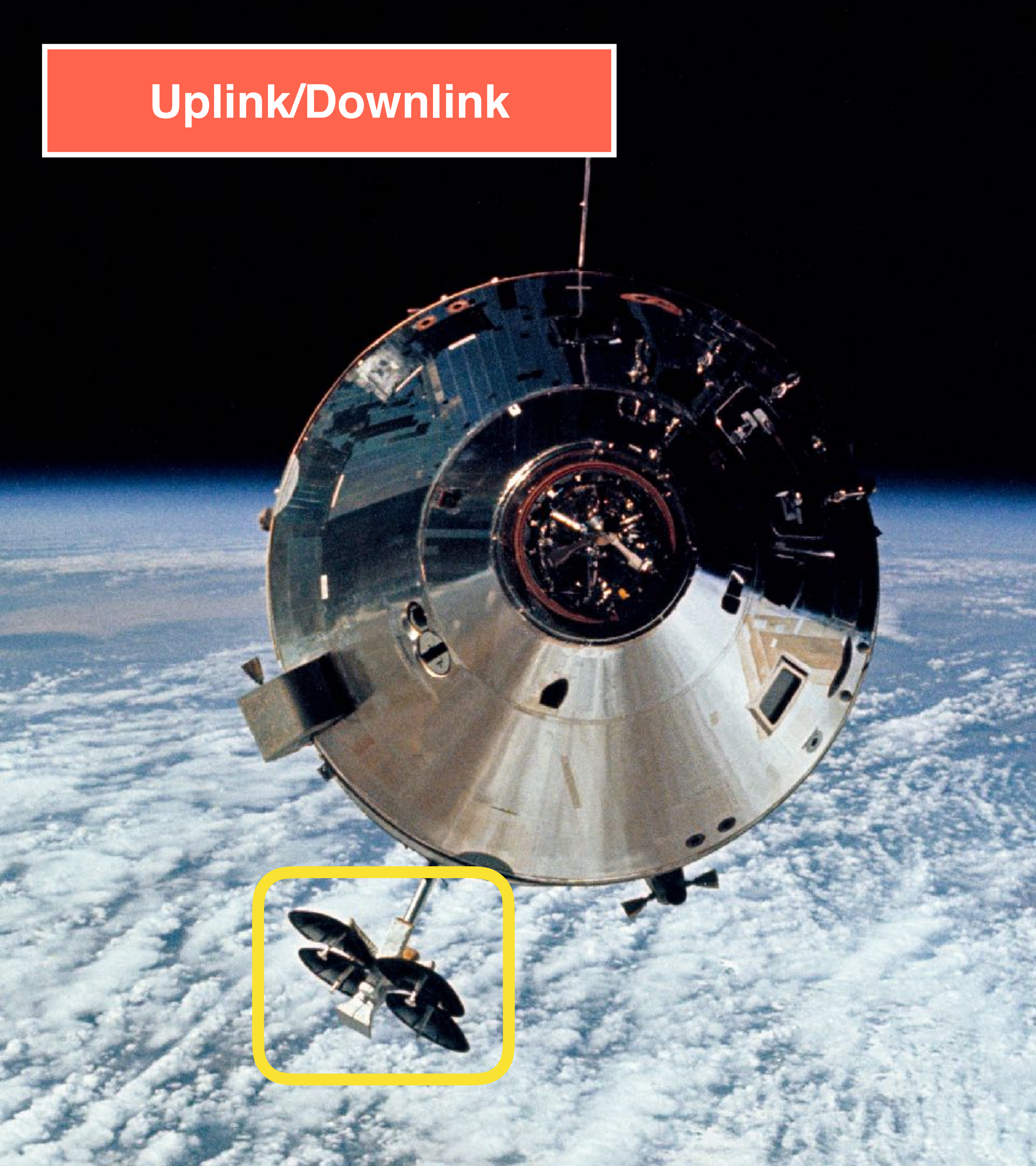
00D

0-4 key code

The Command Module contains two DSKYs, so it has a second set of registers.

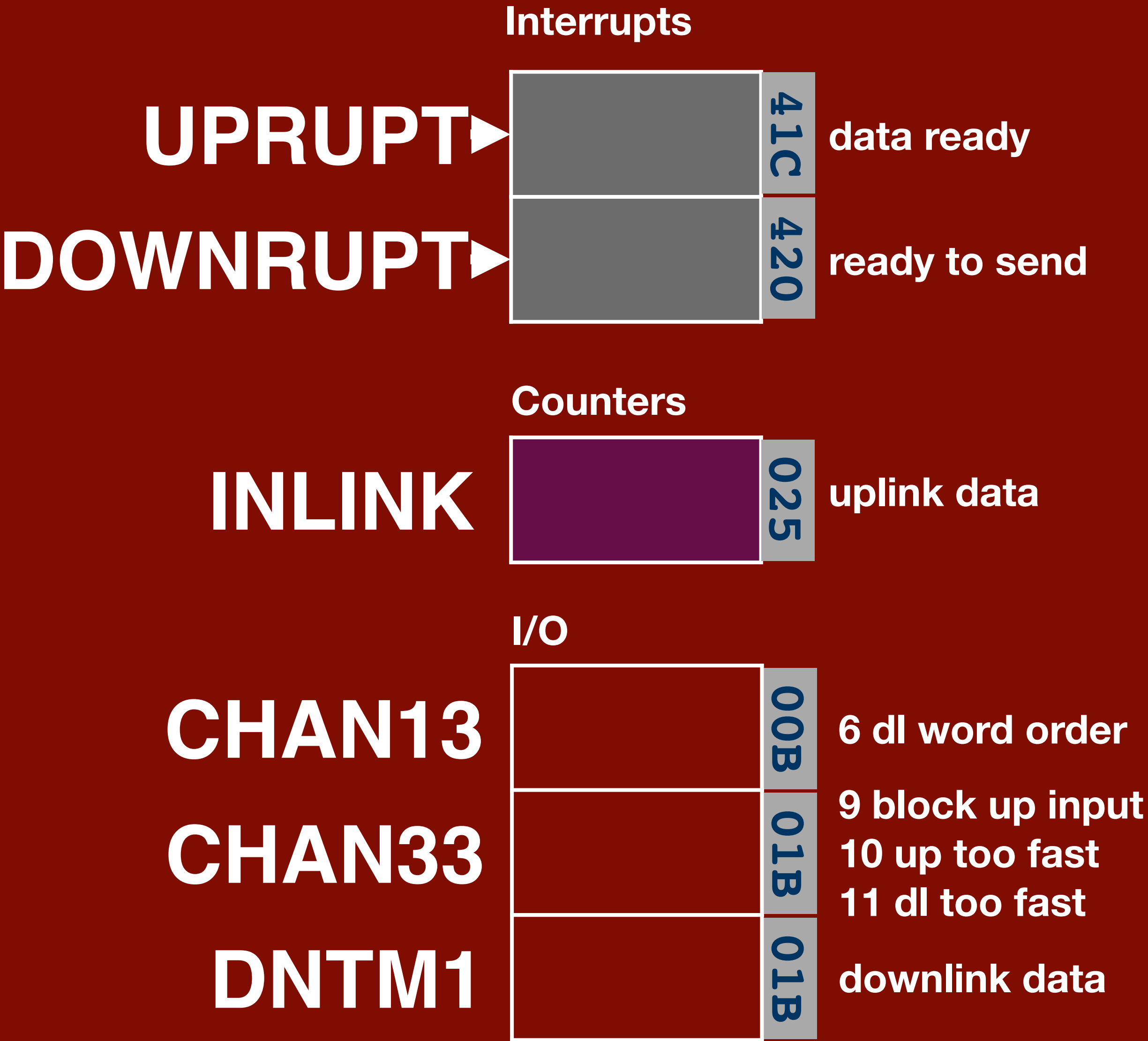
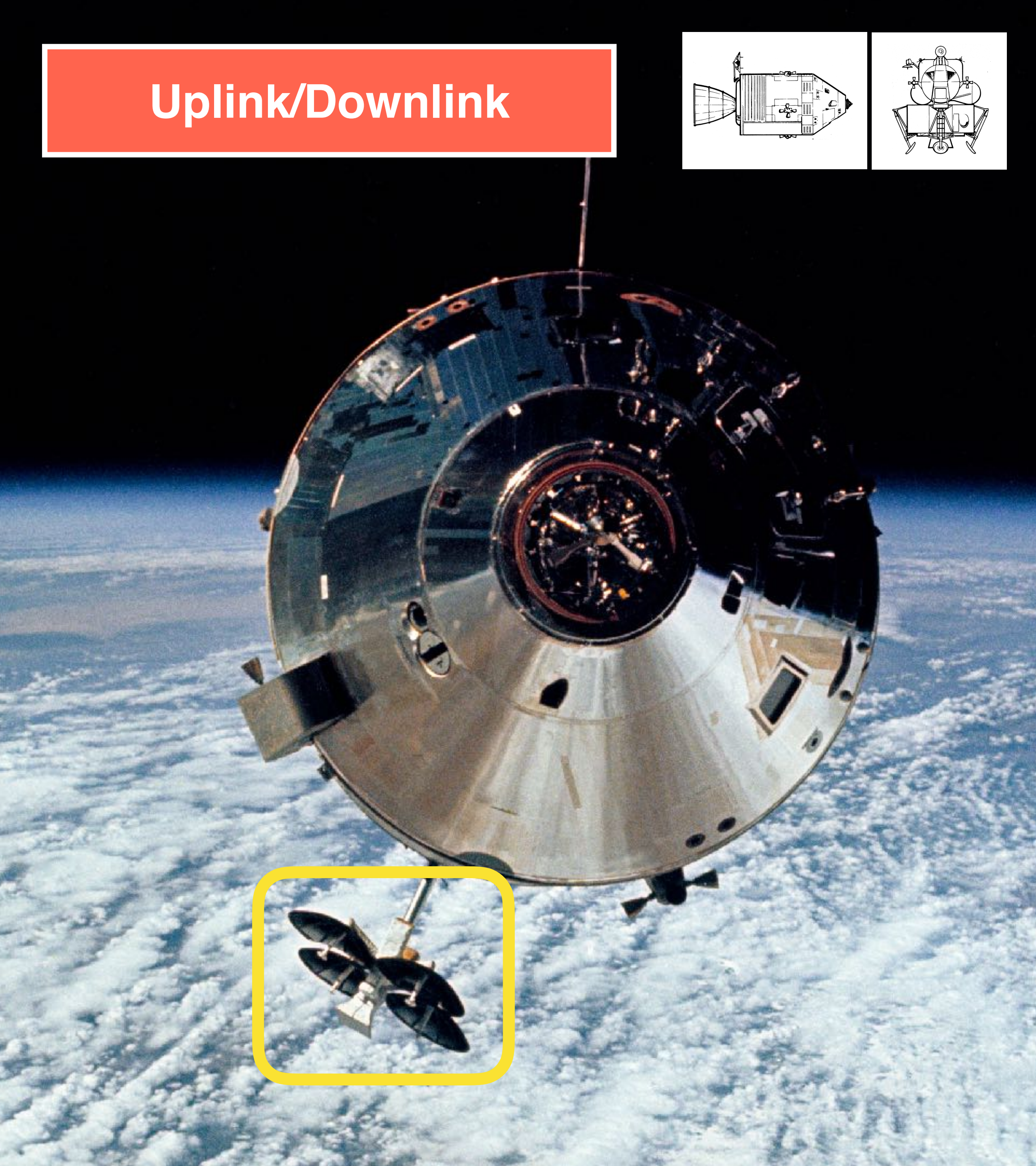
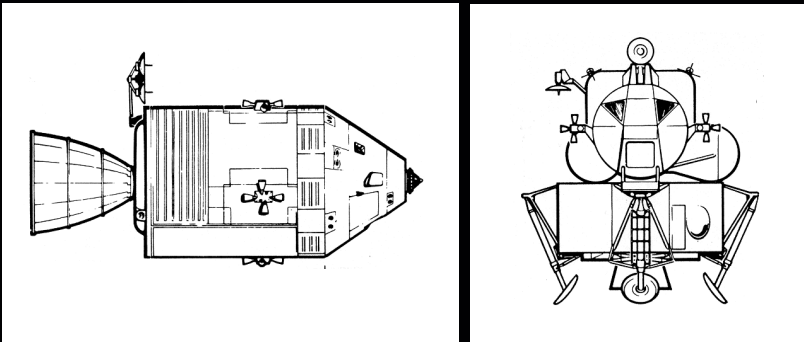


Uplink/Downlink





Uplink/Downlink





## Peripherals

Gyroscope

Landing Radar

DSKY

Accelerometer

Rendezvous Radar

Uplink/Downlink

Optics

RCS Jets



# Peripherals

# Apollo Guidance Computer

Architecture

Mission Software

Hardware

Peripherals

System Software

**System Software**



# System Software

Operating System

Device Drivers

System Utilities

Application Software

Network Software

Security Software

System Management Software

System Configuration Software

System Monitoring Software

System Backup Software

System Recovery Software

System Maintenance Software

System Performance Software

System Security Software

System Administration Software

System Development Software

System Testing Software

**System Software**

**Priority-Based  
Cooperative**

**Preemptive  
Real-Time**

**Interactive**

**Fault-Tolerant**

**Virtual Machines**

# System Software

Operating System

Device Drivers

Firmware



**System Software**

**Multitasking**

**Interpreter**

**Device Drivers**

**Waitlist**

**System Software**

**Multitasking**

**Shell**

**Interpreter**

**Fault Recovery**

**Device Drivers**

**Waitlist**

# Multitasking



**Multitasking**

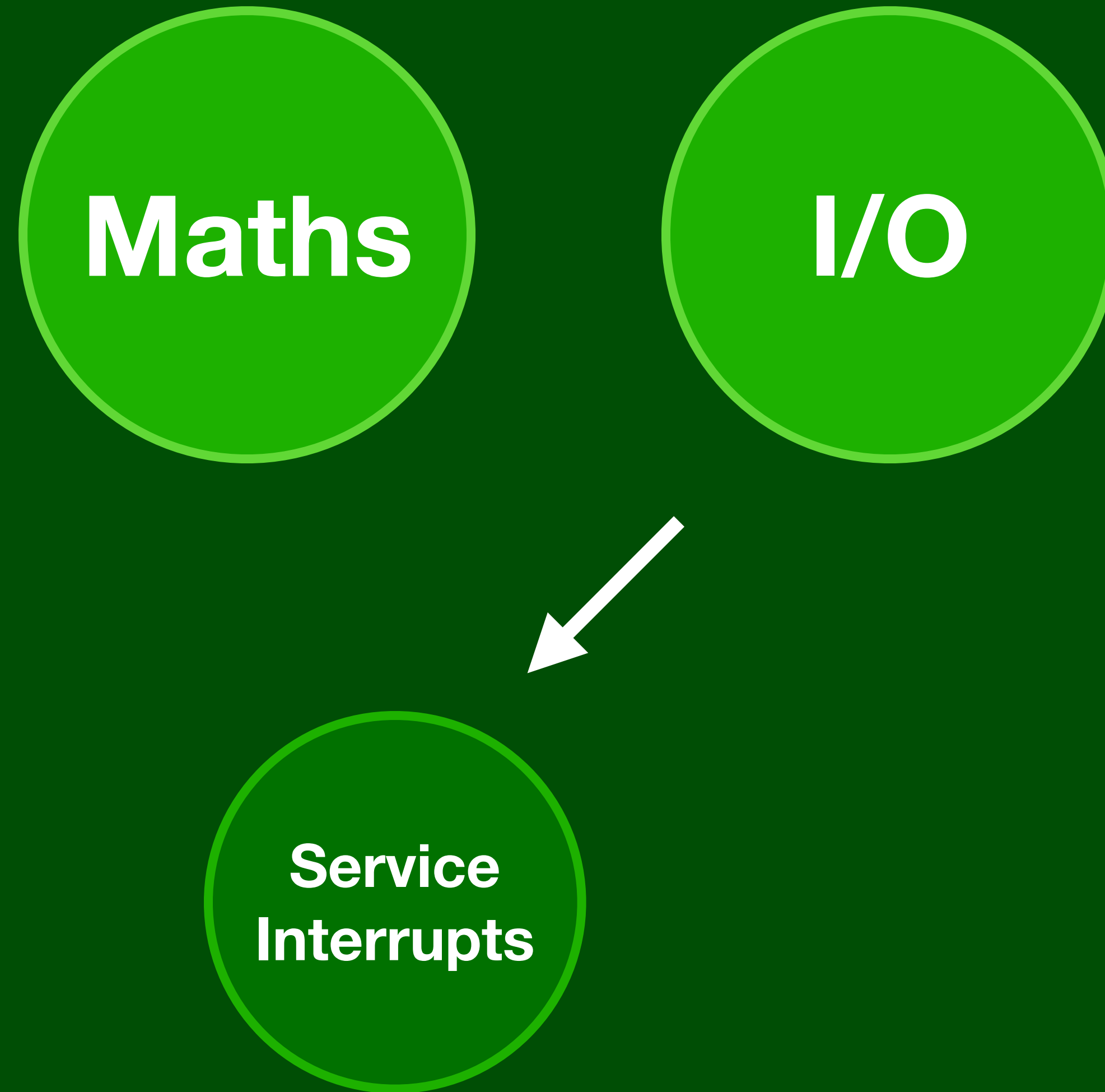
**Maths**

**Multitasking**

**Maths**

**I/O**

# Multitasking



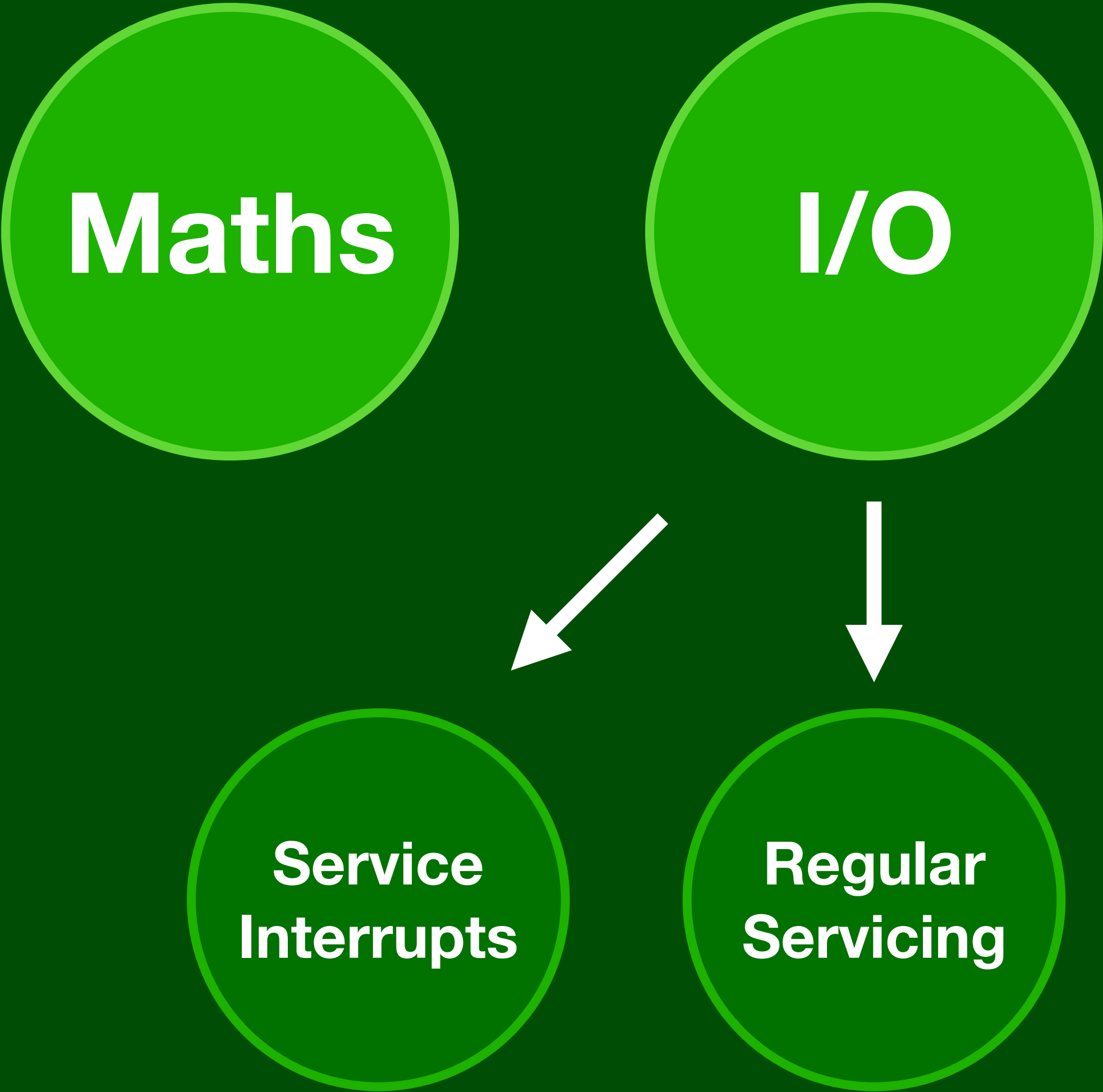
---

**DSKY:**      **Keypress**

---



Multitasking

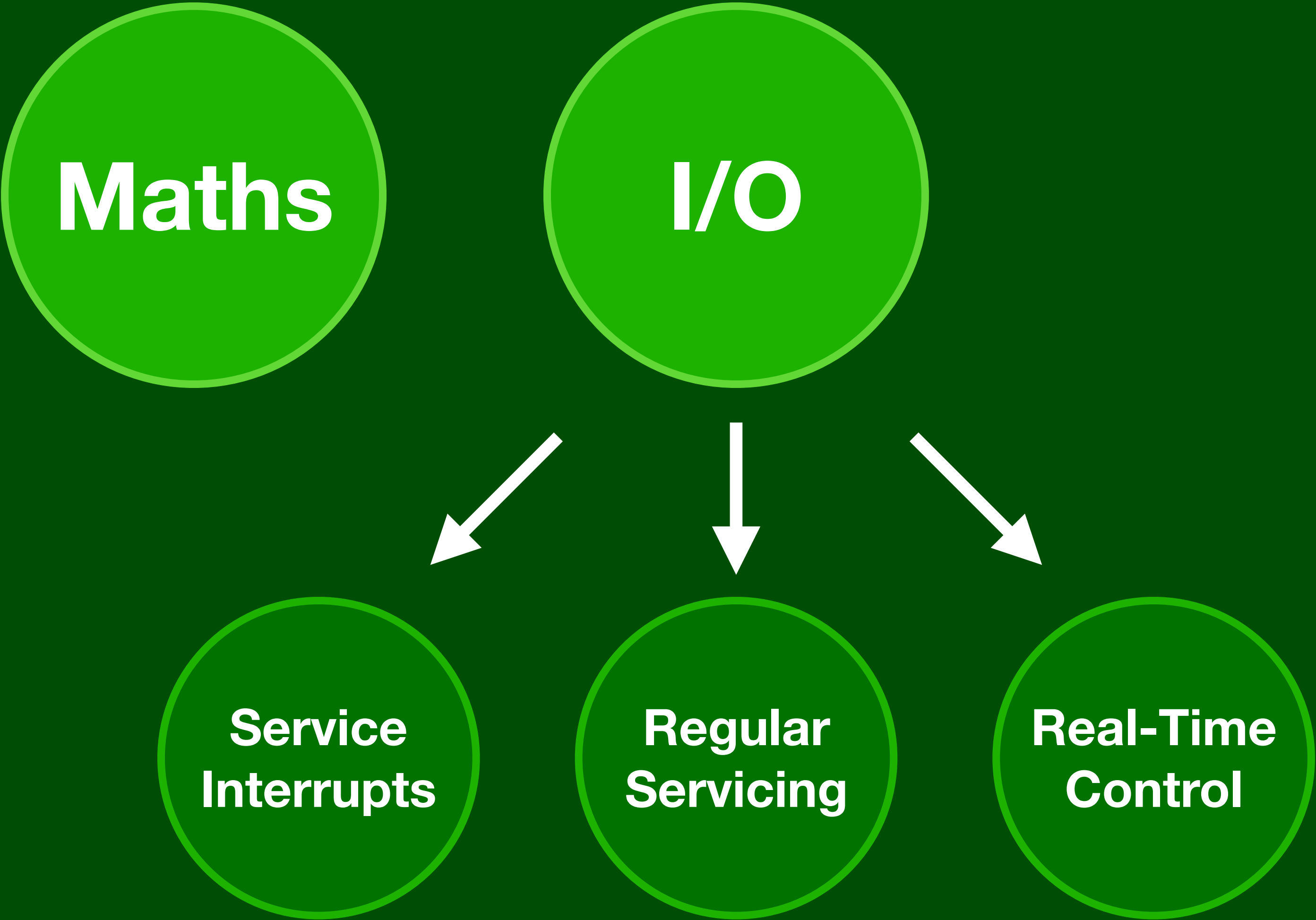


---

DSKY:	Keypress	—	Update Display
-------	----------	---	----------------

---

Multitasking



---

<b>DSKY:</b>	<b>Keypress</b>	<b>—</b>	<b>Update Display</b>	<b>—</b>	<b>Flash Lamp</b>
--------------	-----------------	----------	-----------------------	----------	-------------------

---

# Scheduling



**Scheduling**

**Job**



# Scheduling

Job A

Job B

Job C

Job D



# Scheduling

4 Highest

3 High

2 Medium

1 Low

Job A

Job B

Job C

Job D



# Priority Scheduling

4 Highest

3 High

2 Medium

1 Low

Queue

Running

0

t





# Priority Scheduling

4 Highest

3 High

2 Medium

1 Low

Queue

Job A

Running

Job A

20 ms

0

t



# Priority Scheduling

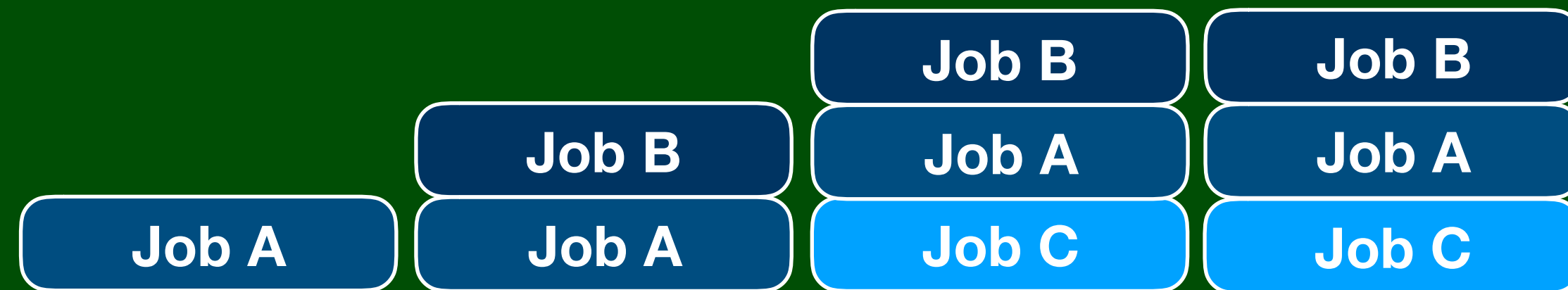
4 Highest

3 High

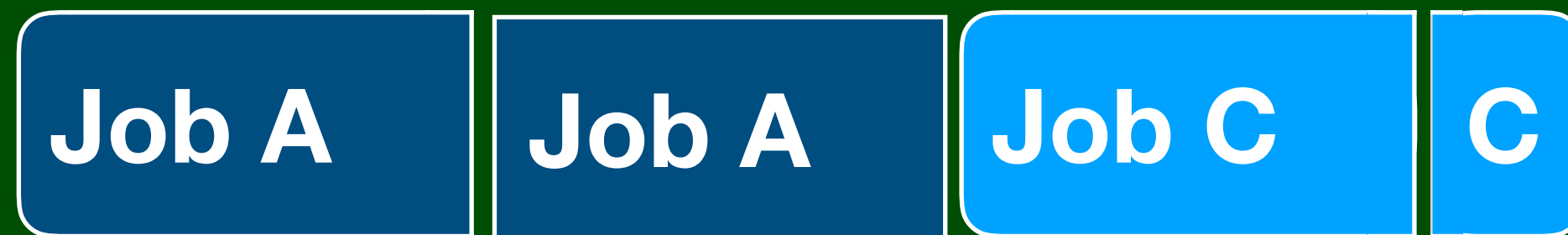
2 Medium

1 Low

Queue



Running



20 ms

0

t

# Priority Scheduling

4 Highest

3 High

2 Medium

1 Low

Queue



Running



20 ms

0

t

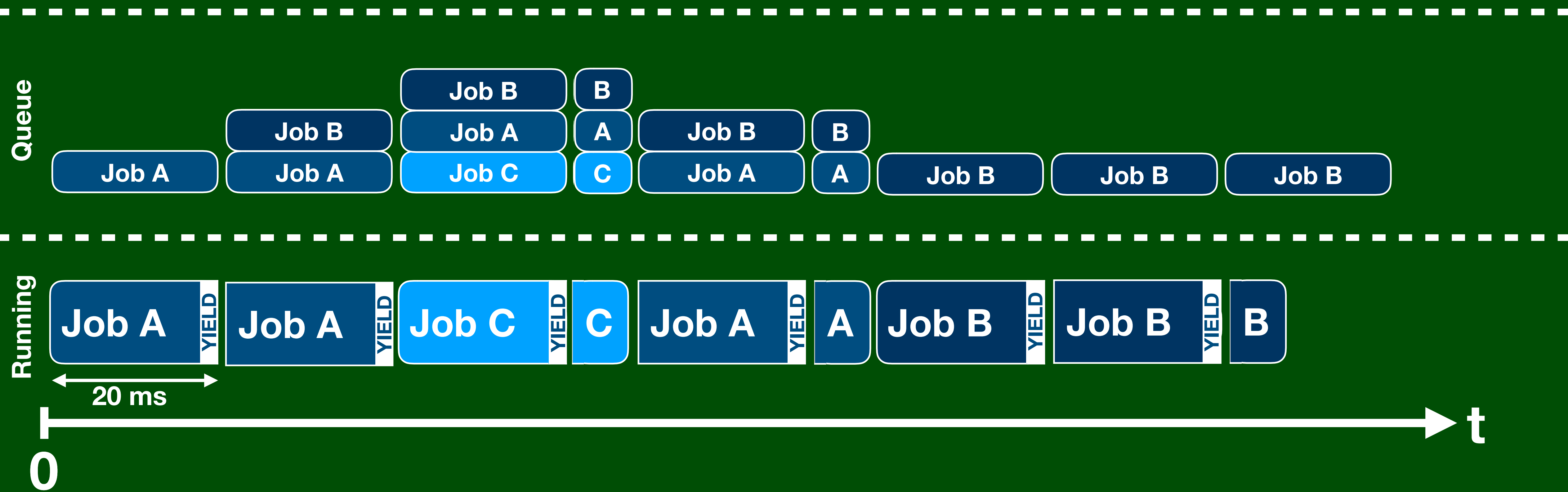
## Priority Scheduling

## 4 Highest

### 3 High

## 2 Medium

**1 Low**





# Priority Scheduling

4 Highest

3 High

2 Medium

1 Low

Queue



Running



20 ms

0

t

**YIELD**

Cooperative  
Multitasking

Job Entry

060	
061	
062	
063	
064	
065	
066	
067	
068	
069	
06A	
06B	

Job Entry

PC	0421	068
	4805	069
BB		06A
	1400	06B

	060
	061
	062
	063
	064
	065
	066
	067
	068
	069
	06A
	06B

Job Entry

disabled

Priority

PC

BB

0421

4805

0

0

1

0

1

0

0-31

060

061

062

063

064

065

066

067

068

069

06A

06B



Job Entry

PC	0421	068
	4805	069
BB		06A
	1400	06B

	060
	061
	062
	063
	064
	065
	066
	067
	068
	069
	06A
	06B

Job Entry

PC	0421	068
	4805	069
BB		06A
	Priority	06B
	+10	

	060
	061
	062
	063
	064
	065
	066
	067
	068
	069
	06A
	06B

Core Set

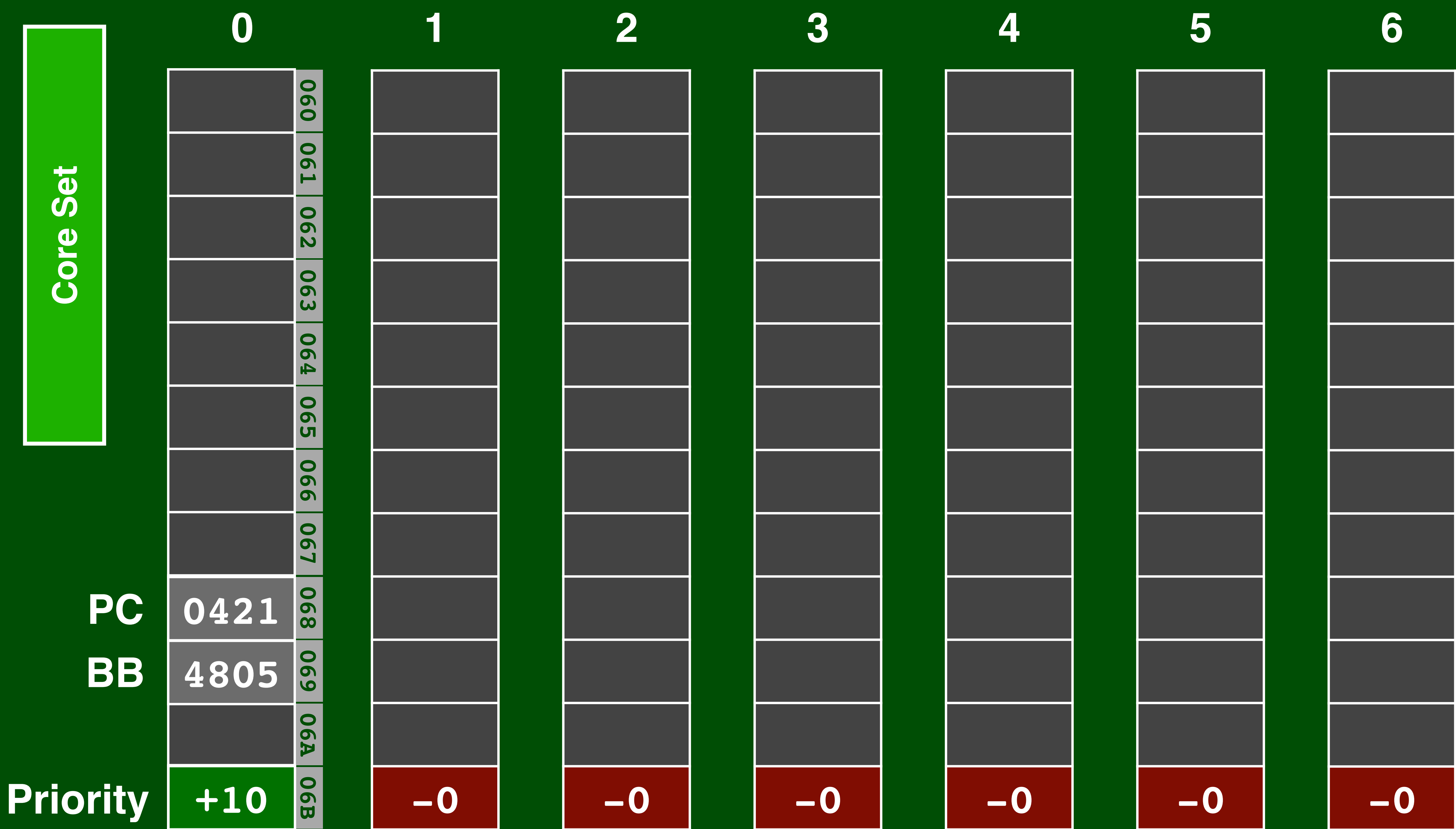
0

PC

BB

Priority

	060
	061
	062
	063
	064
	065
	066
	067
0421	068
4805	069
	06A
+10	06B





## Core Set

060	
061	
062	
063	
064	
065	
066	
067	
068	0421
069	4805
06A	
06B	+10

[illegible][illegible][illegible][illegible][illegible][illegible]

0421

4805

**+10**

**-0**

**-0**

**-0**

**-0**

**-0**

**-0**

## Core Set

060	
061	
062	
063	
064	
065	
066	
067	
068	0421
069	4805
06A	
06B	+10

07A0
4805
+12

[illegible][illegible][illegible][illegible][illegible]

0421

07A0

4805

4805

**+10**

**+12**

**-0**

**-0**

**-0**

**-0**

—0

## Core Set

060	
061	
062	
063	
064	
065	
066	
067	
068	0421
069	4805
06A	
06B	+10

07A0
4805
+12

[illegible][illegible][illegible][illegible][illegible]

0421

07A0

4805

4805

**+10**

**+12**

**-0**

**-0**

**-0**

**-0**

—0

## Core Set

	060
	061
	062
	063
	064
	065
	066
	067
07A0	068
4805	069
	06A
+12	06B

0421
4805
+10

[illegible][illegible][illegible][illegible][illegible]

07A0

4805

+12

+12

+10

**-0**

**-0**

**-0**

**-0**

**-0**



## Core Set

	060
	061
	062
	063
	064
	065
	066
	067
07A0	068
4805	069
	06A
+12	06B

0421
4805
+10

[illegible][illegible][illegible][illegible][illegible]

07A0

# 0421

4805

4805

**+12**

**+10**

**-0**

**-0**

**-0**

**-0**

-0

## Core Set

	060
	061
	062
	063
	064
	065
	066
	067
07A0	068
4805	069
	06A
+12	06B

0421
4805
-10

[illegible][illegible][illegible][illegible][illegible]

07A0

0421

4805

4805

**+12**

**-10**

**-0**

**-0**

**-0**

**-0**

**-0**

## Core Set

	060
	061
	062
	063
	064
	065
	066
	067
07A0	068
4805	069
	06A
+12	06B

0421
4805
+10

[illegible][illegible][illegible][illegible][illegible]

07A0

# 0421

4805

4805

**+12**

**+10**

**-0**

**-0**

**-0**

**-0**

**-0**

## Core Set

	060
	061
	062
	063
	064
	065
	066
	067
07A0	068
4805	069
	06A
+12	06B

0421
4805
+10

[illegible][illegible][illegible][illegible][illegible]

07A0

0421

4805

4805

**+12**

**+10**

**-0**

**-0**

**-0**

—0

-0



Core Set

0

1

2

3

4

5

6

Local Storage

060 061 062 063 064 065 066 067 068 069 06A 06B

PC

BB

07A0

4805

0421

4805

+12

+10

-0

-0

-0

-0

-0

Priority

Core Set

0

1

2

3

4

5

6

Local Storage

060 061 062 063 064 065 066 067 068 069 06A 06B

PC

BB

07A0

4805

0421

4805

+12

+10

-0

-0

-0

-0

-0

Priority

# Executive Functions

060	061	062	063	064	065	066	067	068	069	06A	06B
Local Storage								07A0	4805		+12

# Local Storage

060	061	062	063	064	065	066	067	068	069	06A	06B
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----

07A0

4805

+12

07A0

4805

+12

1
0421
4805
+10

0421

4805

+10

[illegible]

**-0**

[illegible]

**-0**

[illegible]

**-0**

[illegible]

**-0**

[illegible]

**-0**

Executive Functions

0

Local Storage		060
		061
		062
		063
		064
		065
		066
		067
		068
		069
PC	07A0	06A
BB	4805	06B
Priority	+12	

1

0421
4805
+10

2

NOVAC
-0

3

Create new job
-0

4

-0

5

-0

6

-0



# Executive Functions

060	061	062	063	064	065	066	067	068	069	06A	06B
Local Storage								07A0	4805		+12

# Local Storage

Local Storage

07A0

4805

+12

07A0

4805

06A

+12

+12

# OVAC RIOCH

Cre  
Cha

new job  
priority

current  $j$

— 0

# Create new job

# Change priority of current job

# Priority

# Executive Functions

060	061	062	063	064	065	066	067	07A0	4805		+12
Local Storage											

# Local Storage

A vertical diagram representing memory segments. It consists of four stacked rectangular blocks. The top block is orange and contains the text "Local Storage". The second block is grey and contains the text "07A0". The third block is grey and contains the text "4805". The bottom block is green and contains the text "+12".

07A0

4805

\_\_\_\_\_

+12

# OVAC RIOCH OBSLE

- Create
- Change
- Put
- -0

new job  
priority  
rent job

current j  
sleep

# Create new job

# Change priority of current job

# Put current job to sleep

# Priority

# Executive Functions

060	061	062	063	064	065	066	067	07A0	4805		+12
Local Storage											

# Local Storage

A vertical diagram representing memory segments. It consists of four stacked rectangular blocks. The top block is orange and contains the text "Local Storage" in white. The second block is grey and contains the text "07A0" in white. The third block is grey and contains the text "4805" in white. The bottom block is green and contains the text "+12" in white. A green vertical bar is on the left side of the diagram.

07A0

4805

06A

+12

+12

# OVAC RIOCH OBSLE OBWAK

Create  
 Change  
 Put  
 Wake

new job  
priority  
ent job  
job

current j  
sleep

—0

# Create new job

# Change priority of current job

# Put current job to sleep

# Wake a job

Executive Functions

0

Local Storage		060
		061
		062
		063
		064
		065
		066
		067
PC	07A0	068
BB	4805	069
		06A
Priority	+12	06B

1

0421
4805
+10

2

-0

3

-0

4

-0

5

-0

6

-0

NOVAC

Create new job

PRIOCHNG

Change priority of current job

JOBSLEEP

Put current job to sleep

JOBWAKE

Wake a job

ENDOFJOB

End current job



**Yield**

Yield

YIELD			
	1037	41F	ccs NEWJOB
	0A52	420	call CHANG1
	3103	421	ld a, [\$103]

Yield

YIELD	0000	E	
	1037	41F	ccs NEWJOB
	0A52	420	call CHANG1
	3103	421	ld a, [\$103]
		42	

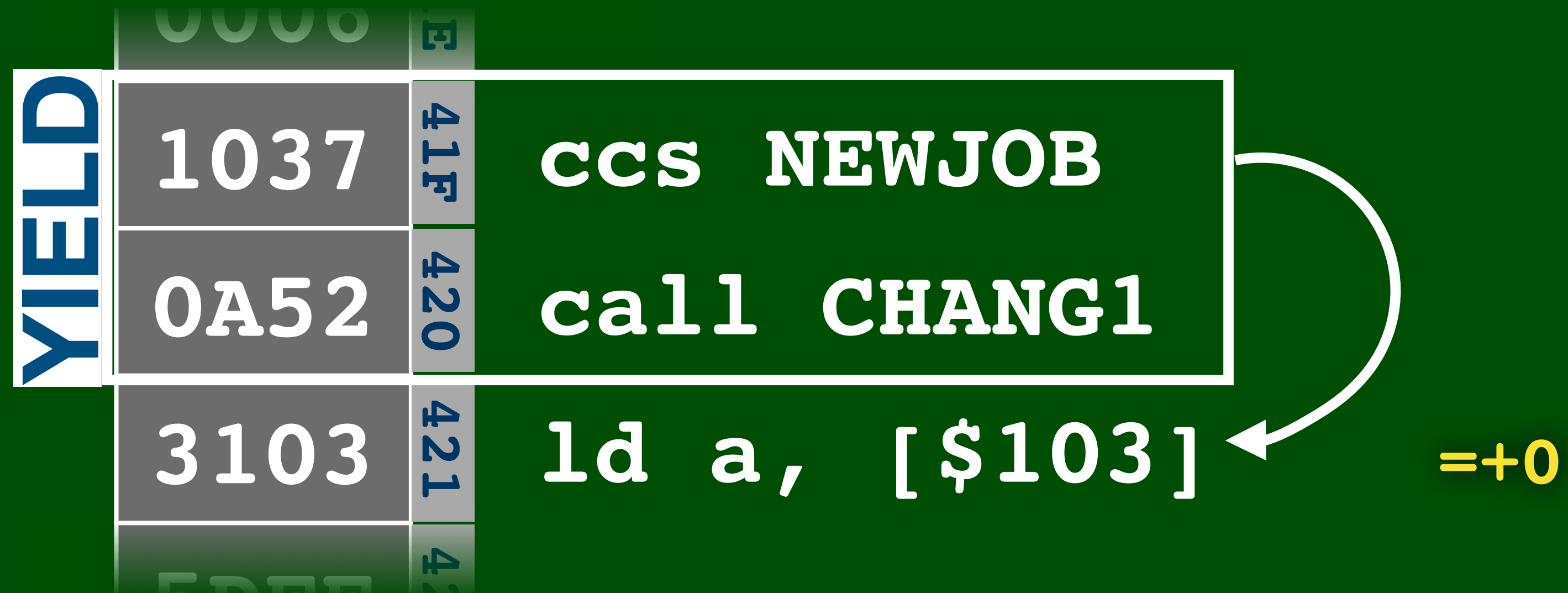
NEWJOB

0000

037

Job with Highest Priority

Yield



NEWJOB

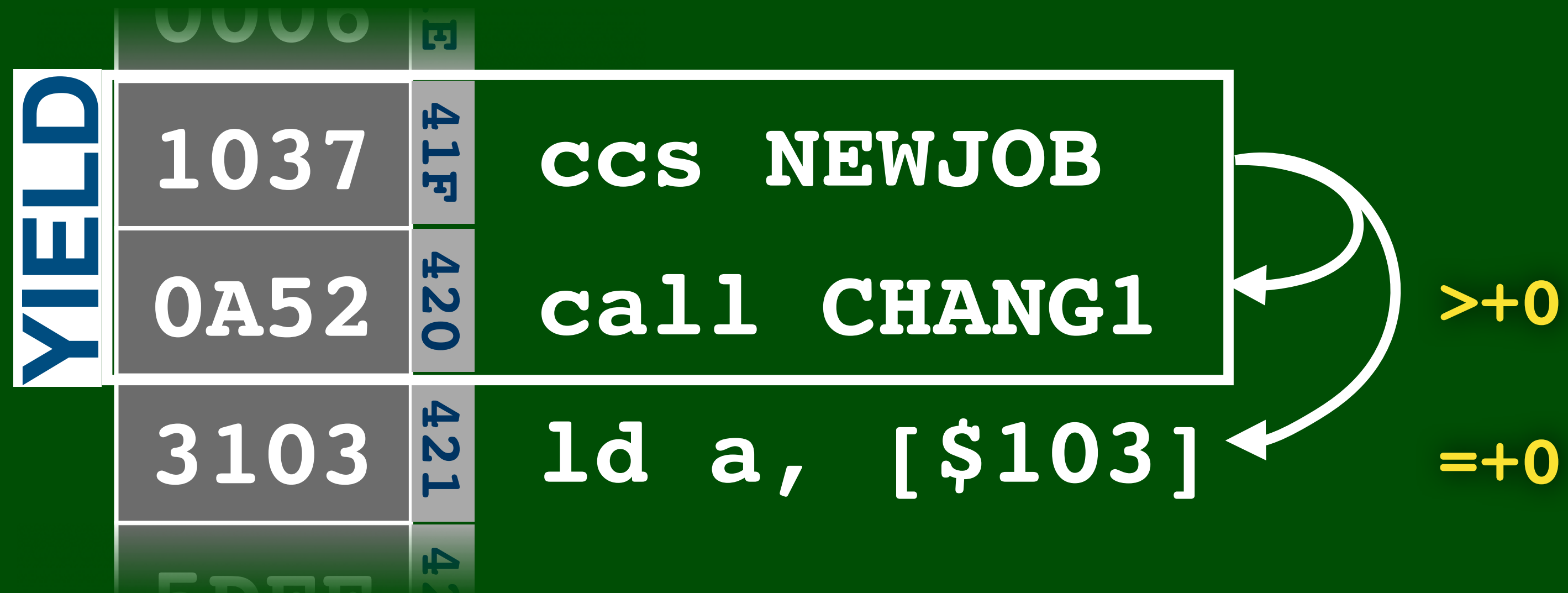
0000

037

Job with Highest Priority



Yield



NEWJOB

0000

037

Job with Highest Priority

Yield

YIELD	0000	E
	1037	41F
	0A52	420
	3103	421
	52FF	42

ccs NEWJOB

call CHANG1

ld a, [\$103]

>+0

=+0

NEWJOB

0000

037

Job with Highest Priority

8

Shadow

\$10

Editing

\$14

Counters

\$31

Watchdog

# Math Support

Math Support

$$a \cdot \vec{v}_1 + b \cdot \vec{v}_2$$



Math Support

Single



.1234

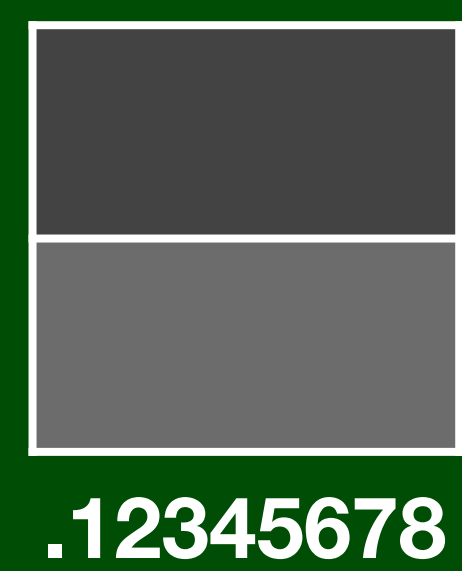
$$a \cdot \vec{v_1} + b \cdot \vec{v_2}$$

Math Support

Single



Double



$$a \cdot \vec{v_1} + b \cdot \vec{v_2}$$

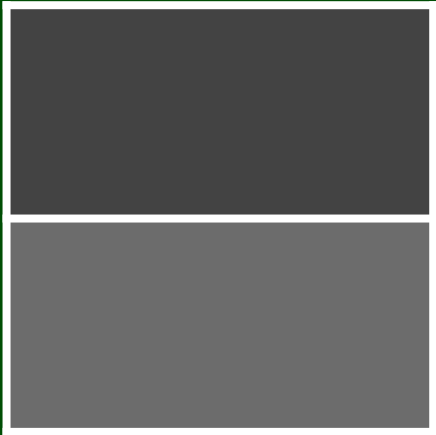
Math Support

Single



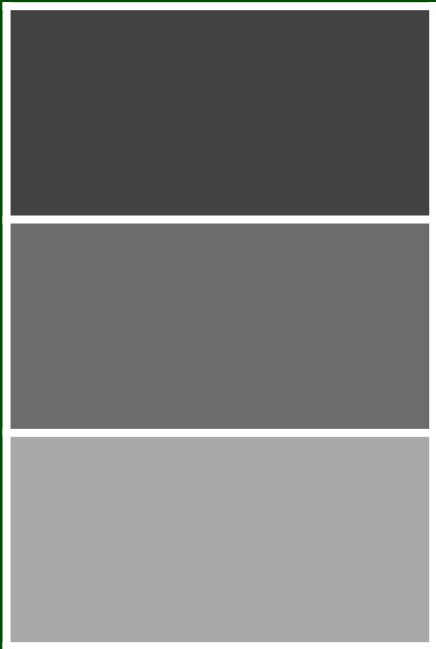
.1234

Double



.12345678

Triple



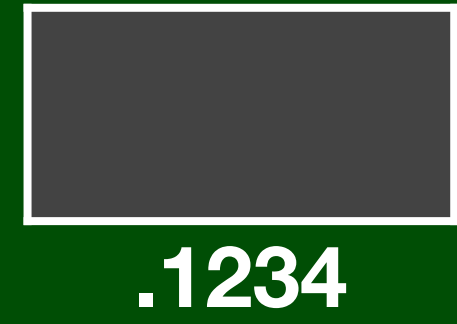
.12345678012

$$a \cdot \vec{v_1} + b \cdot \vec{v_2}$$

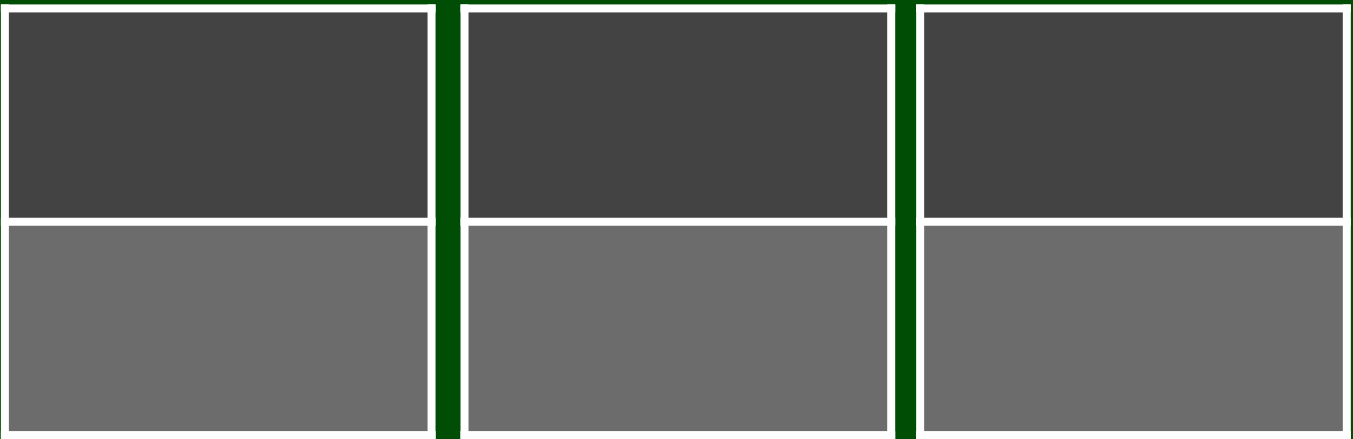
Math Support

$$a \cdot \vec{v_1} + b \cdot \vec{v_2}$$

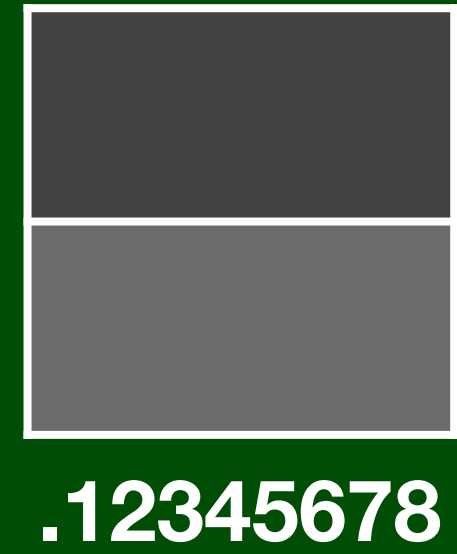
Single



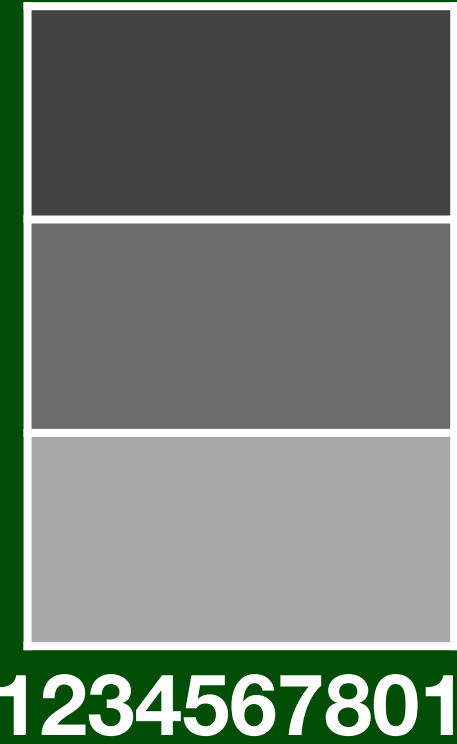
Vector



Double



Triple





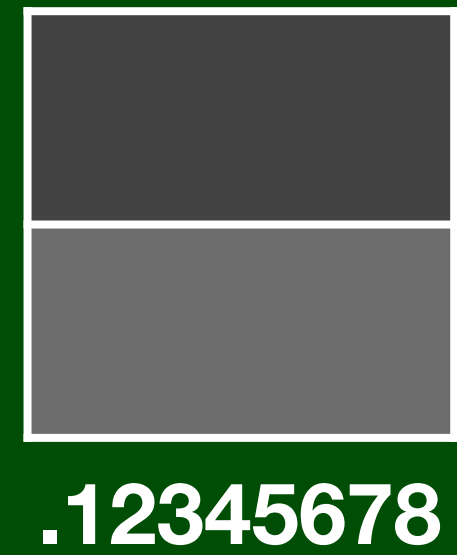
Math Support

$$a \cdot \vec{v_1} + b \cdot \vec{v_2}$$

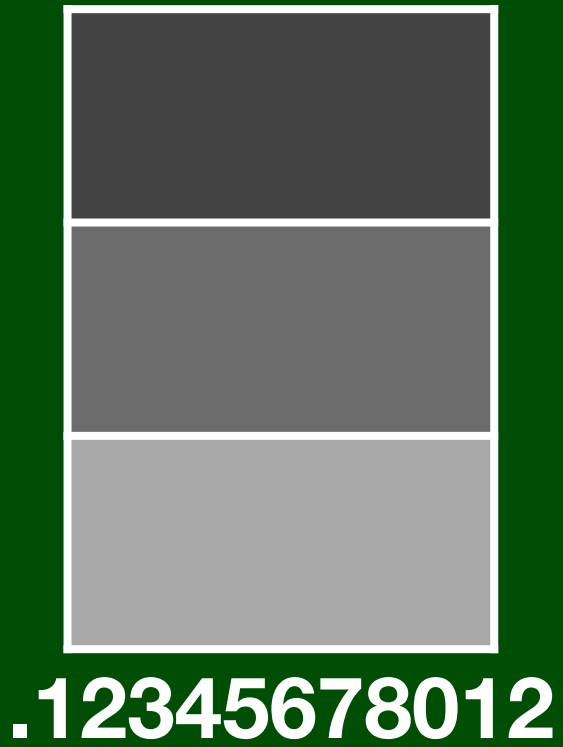
Single



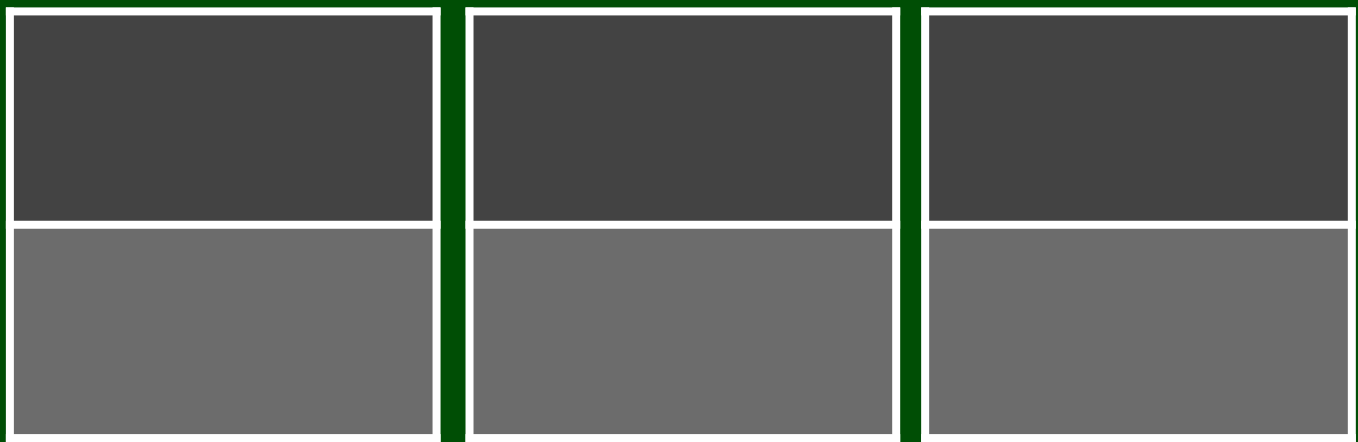
Double



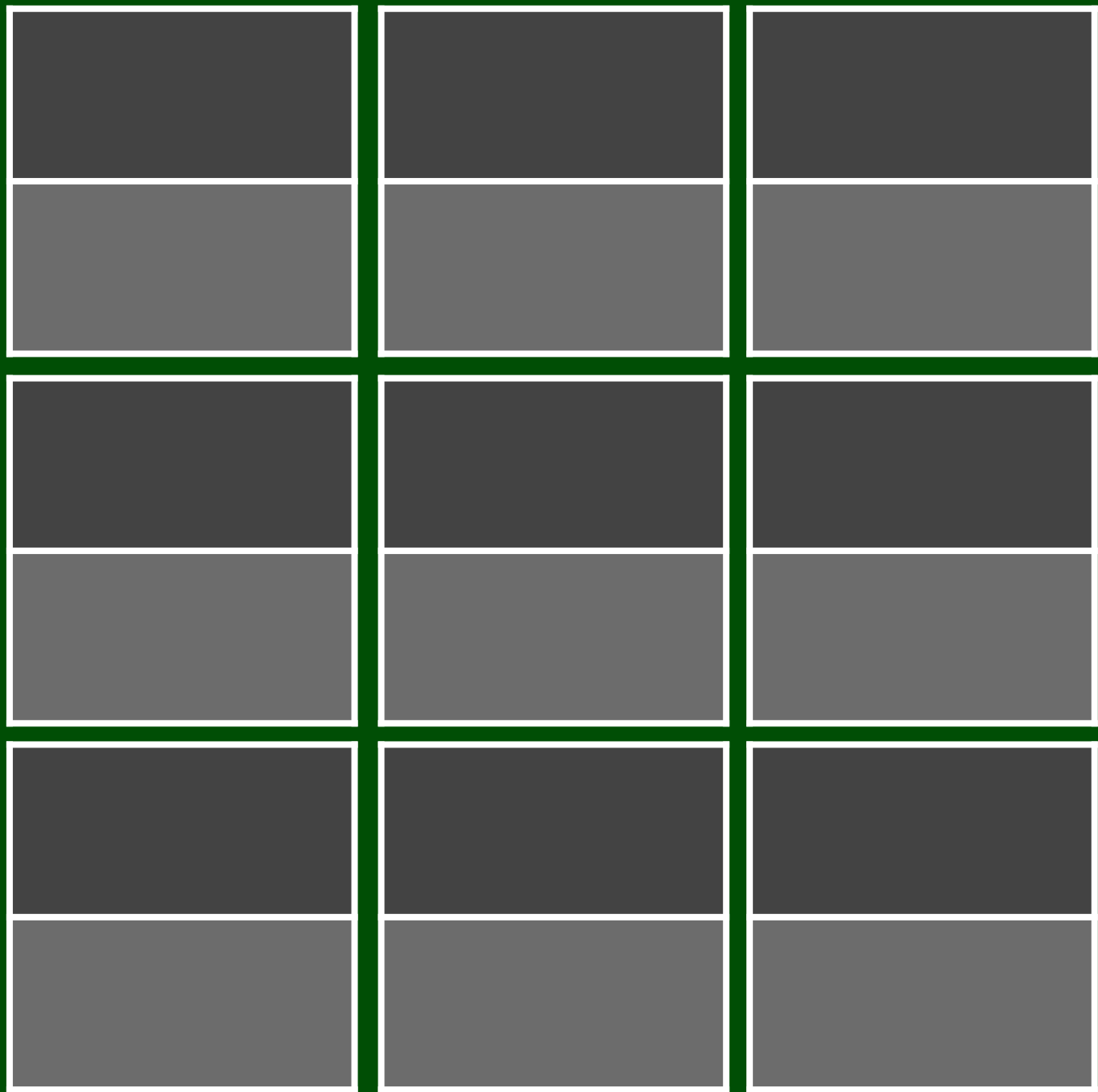
Triple



Vector

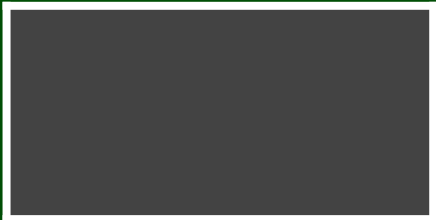


Matrix



Math Support

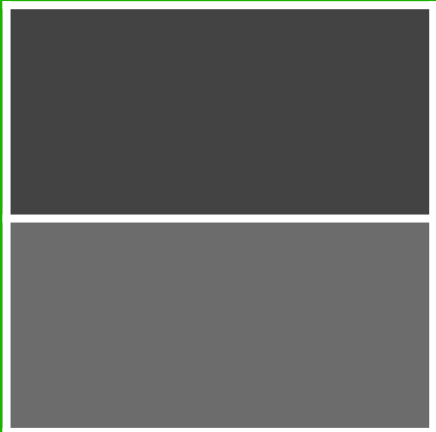
Single



.1234

+0

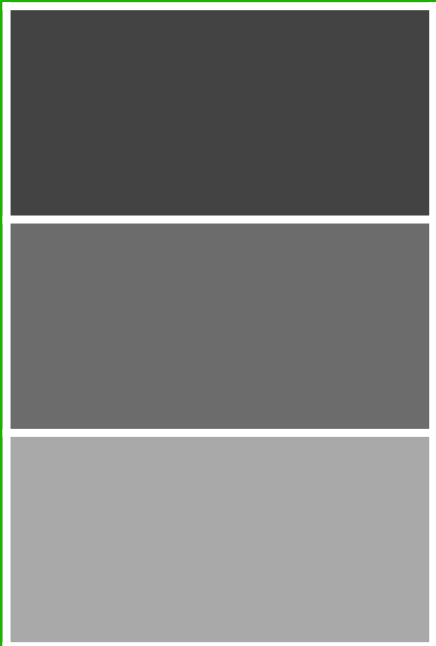
Double



.12345678

+1

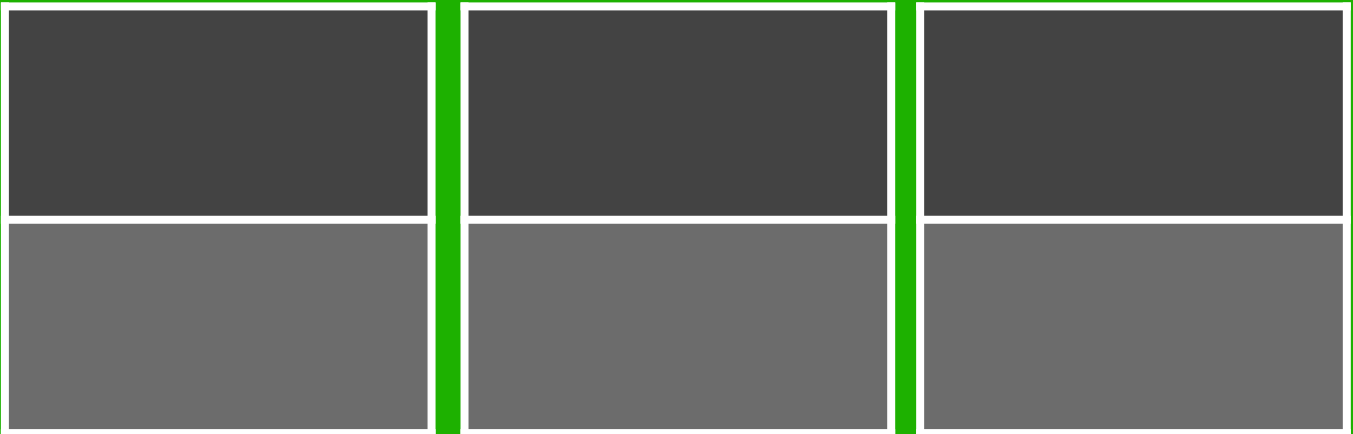
Triple



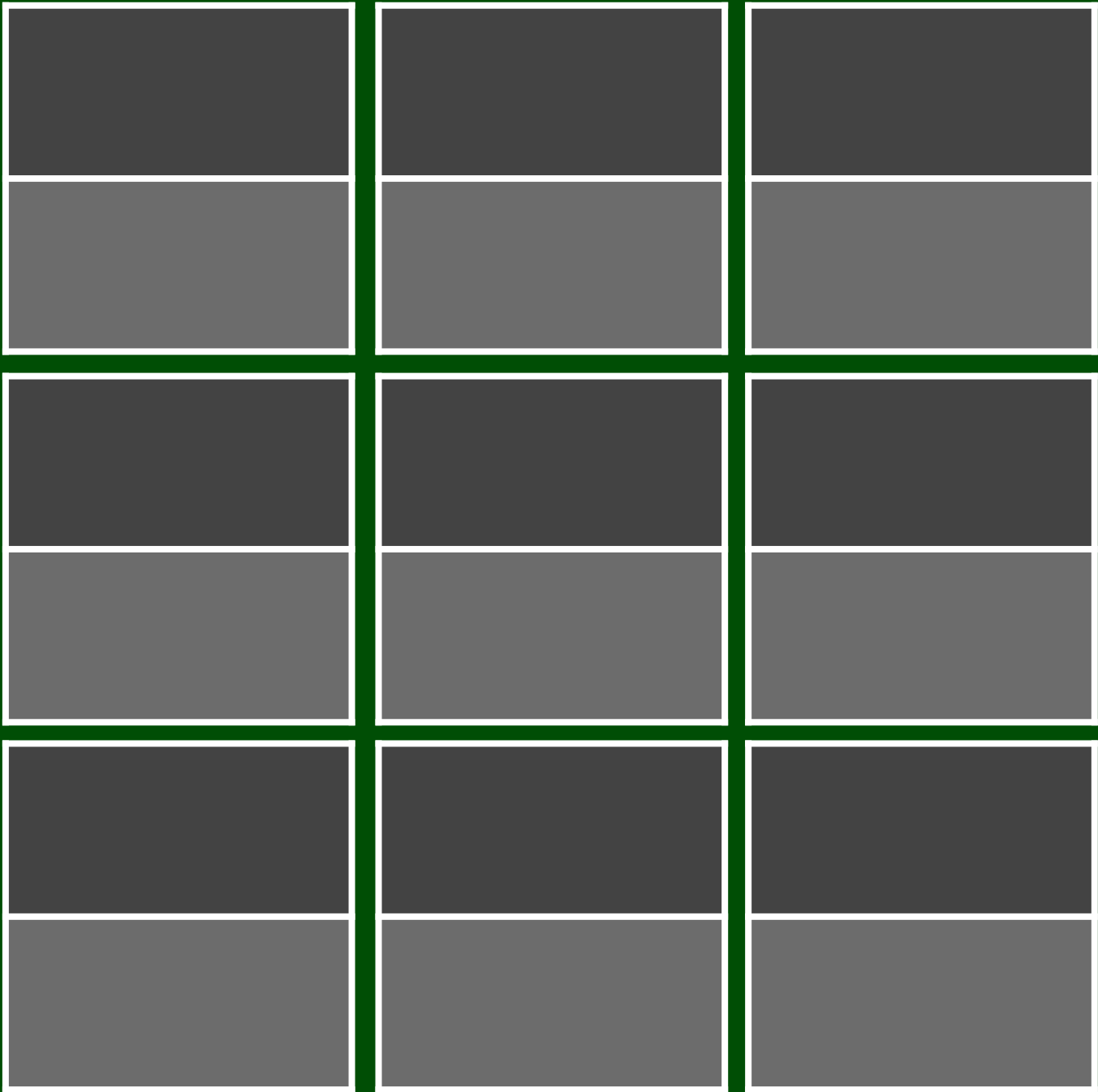
.12345678012

Vector

-1

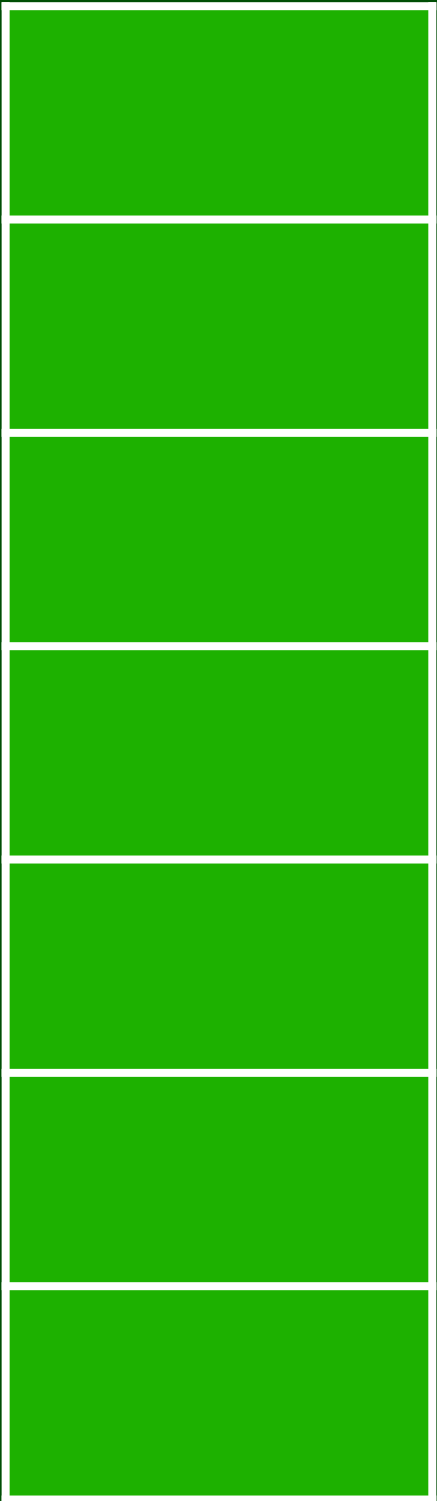


Matrix



$a \cdot \vec{v_1} + b \cdot \vec{v_2}$

MPAC

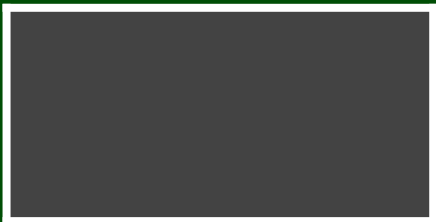


MODE



Math Support

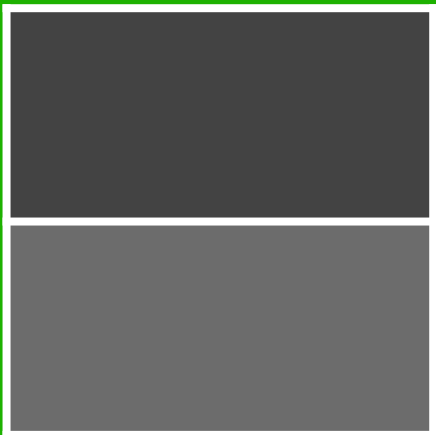
Single



.1234

+0

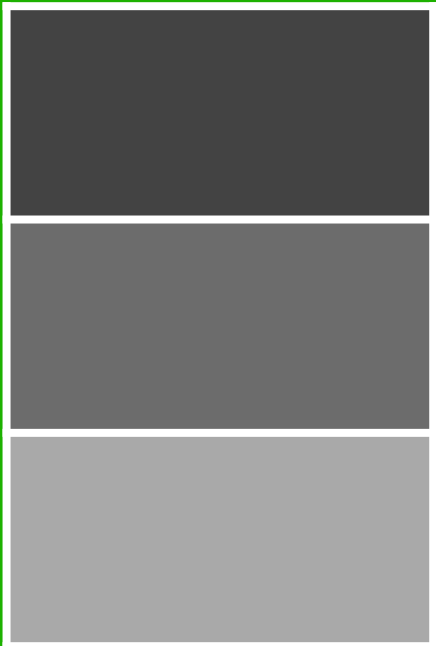
Double



.12345678

+1

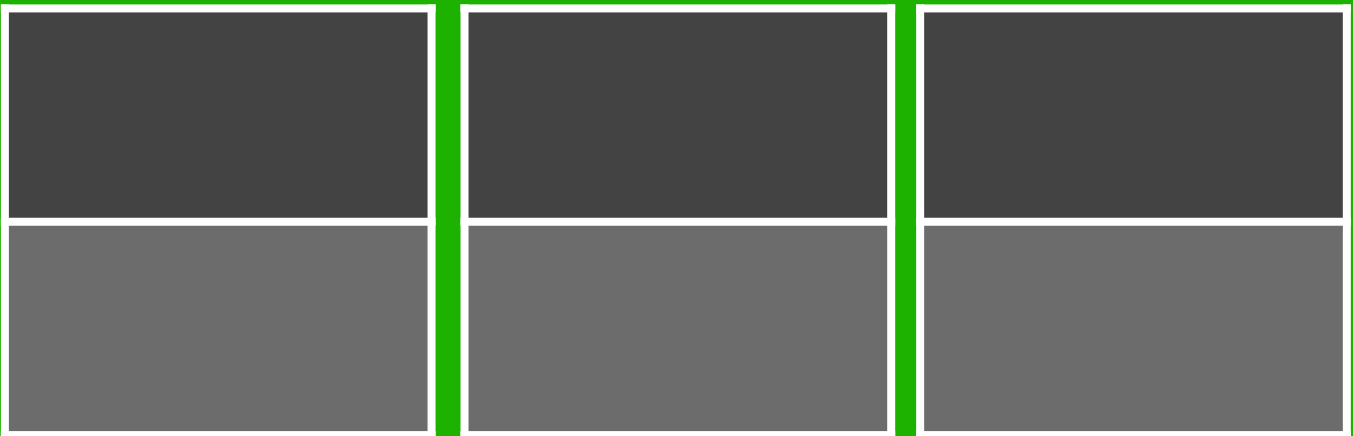
Triple



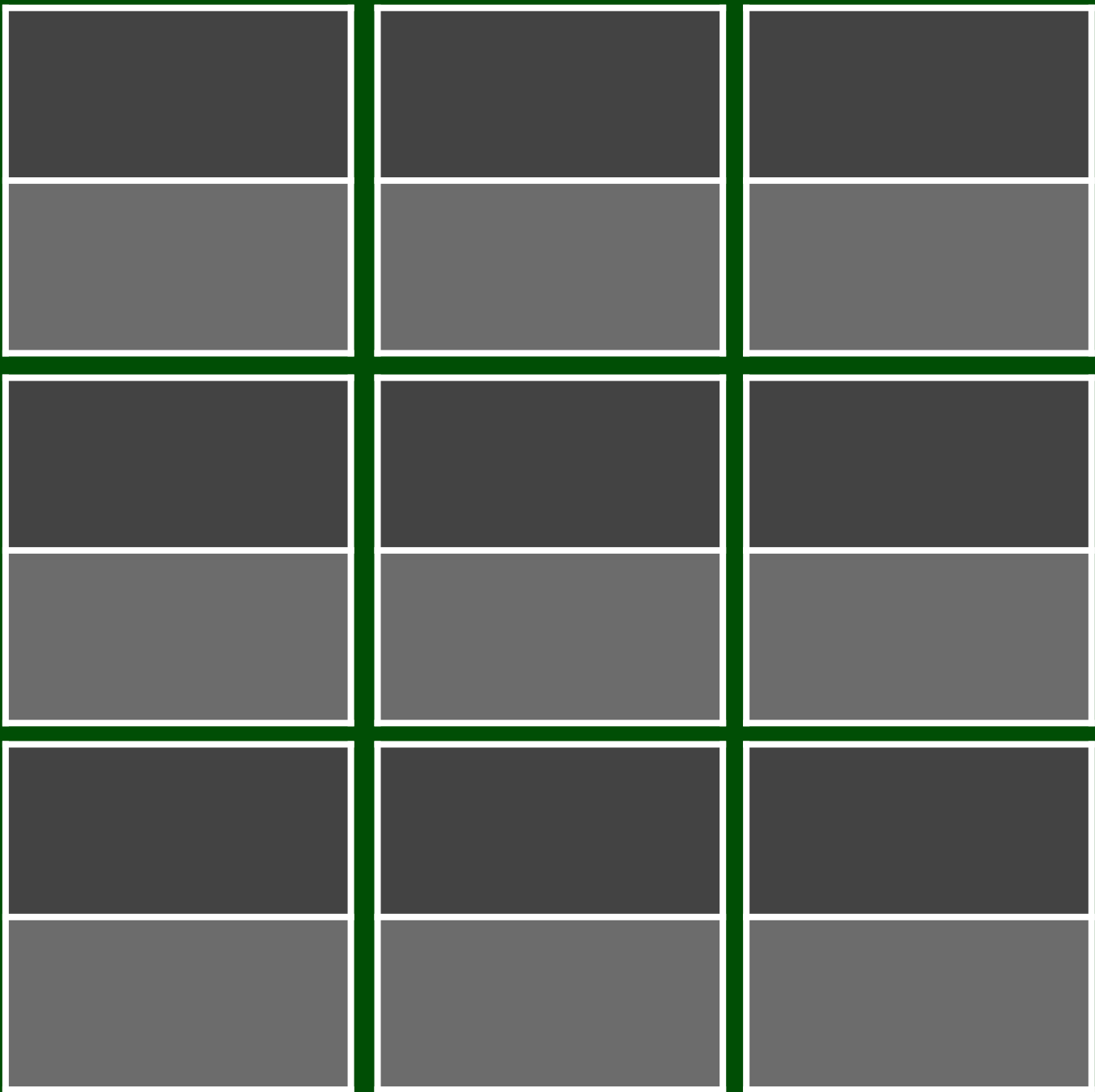
.12345678012

Vector

-1



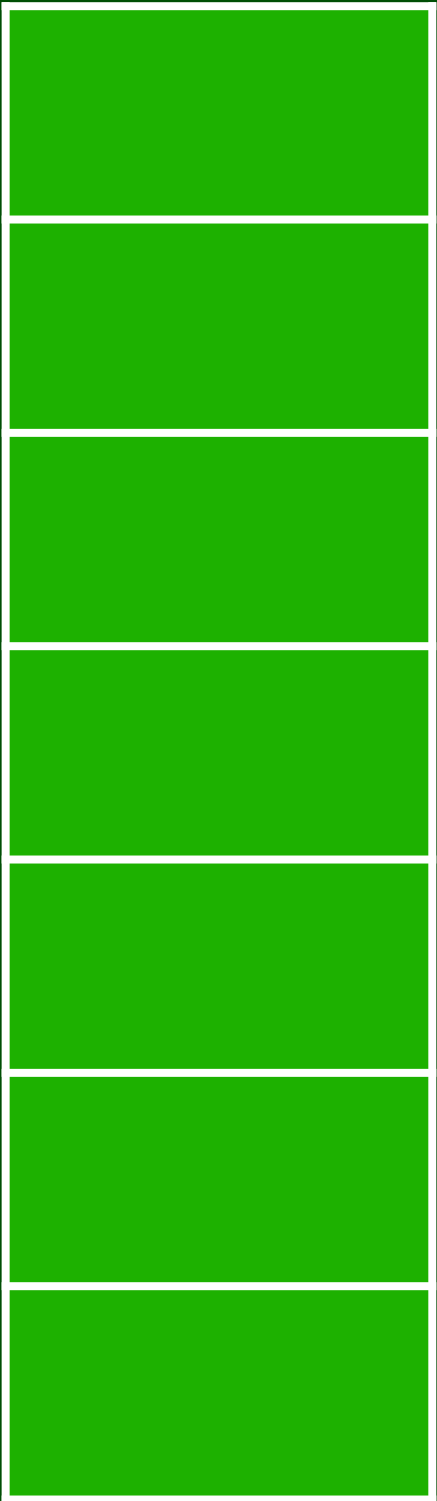
Matrix



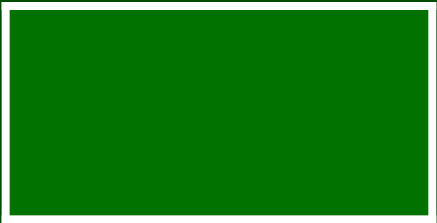
$$a \cdot \vec{v_1} + b \cdot \vec{v_2}$$

```
load_vector(vector_t *v1);
mul_double(double_t *a);
push();
load_vector(vector_t *v2);
mul_double(double_t *b);
add_double(pop());
```

MPAC

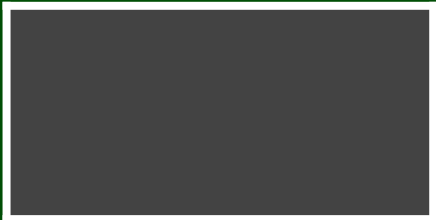


MODE



Math Support

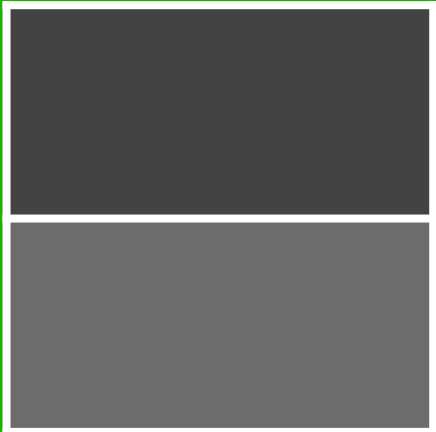
Single



.1234

+0

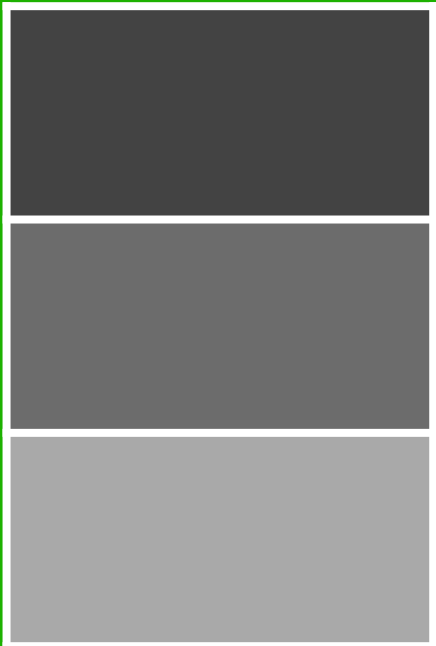
Double



.12345678

+1

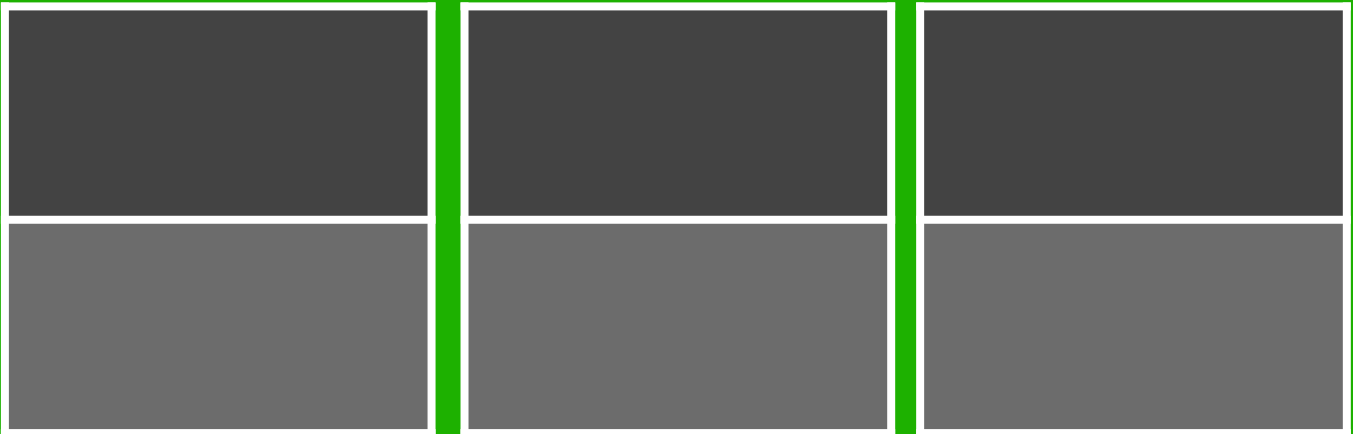
Triple



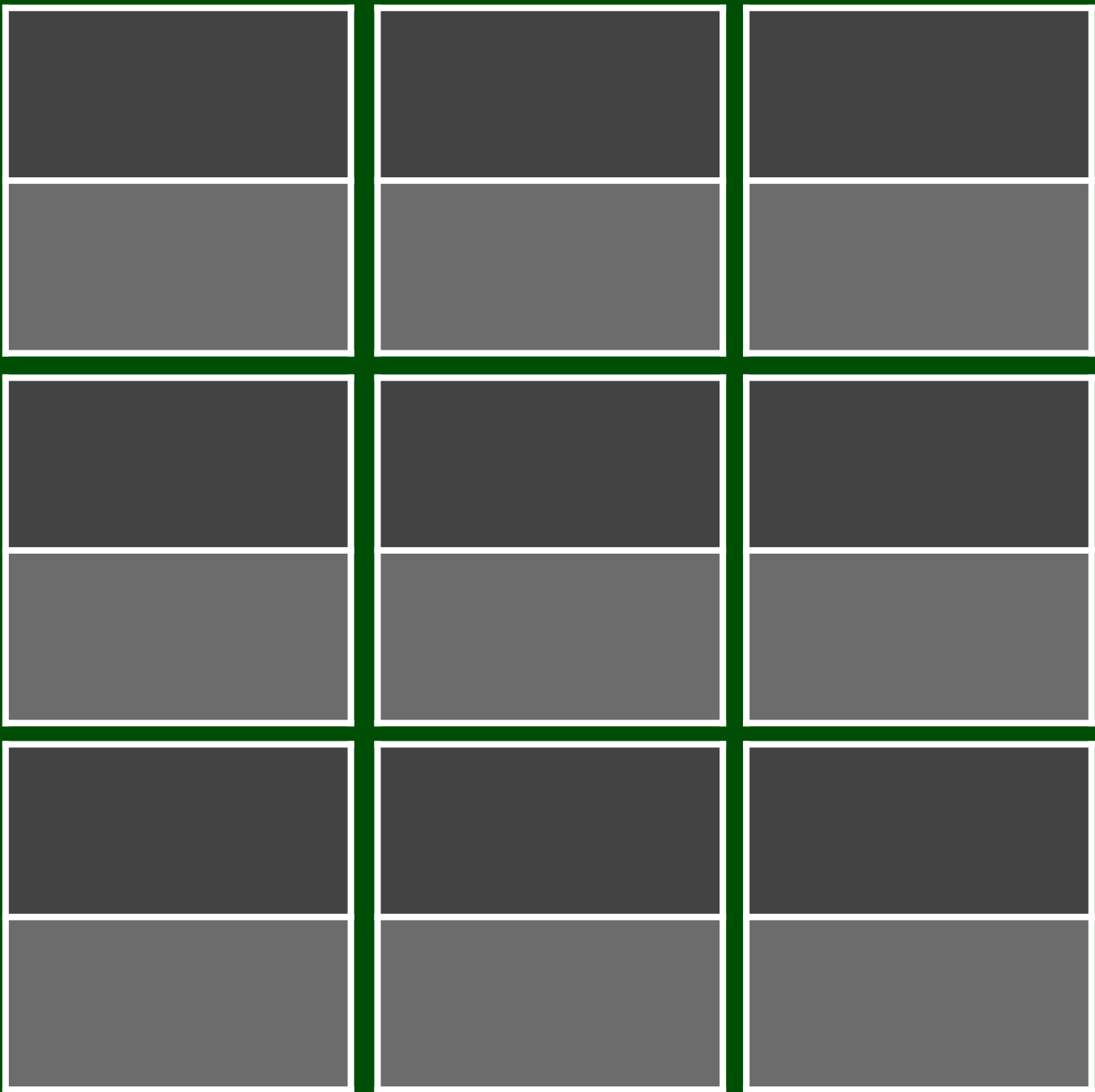
.12345678012

Vector

-1



Matrix

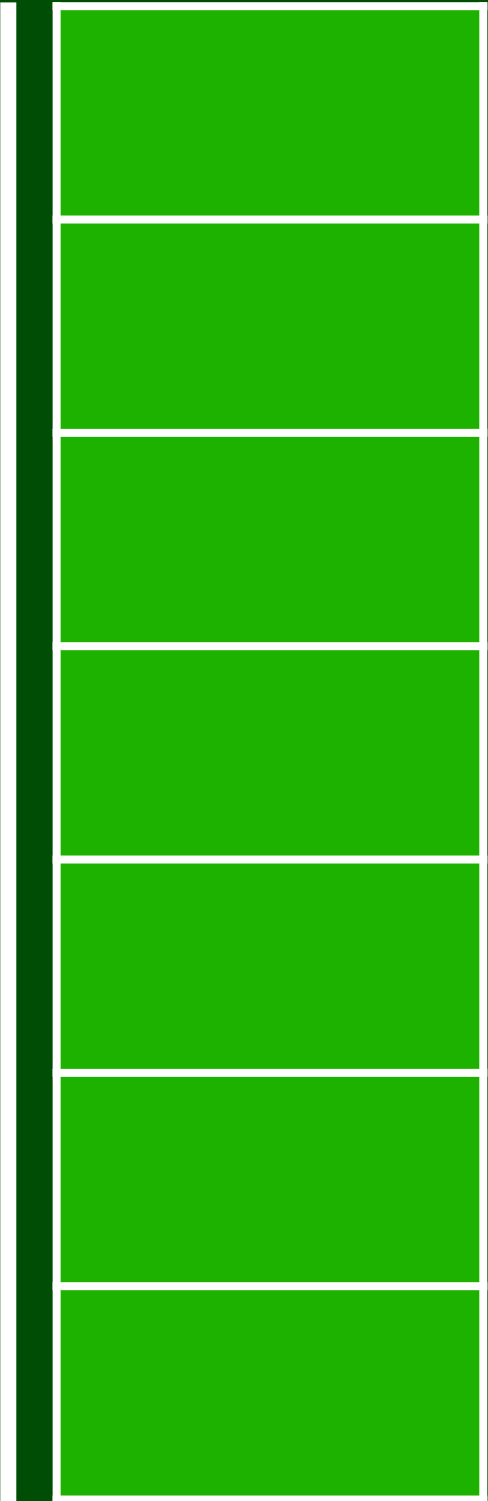


$a \cdot \vec{v_1} + b \cdot \vec{v_2}$

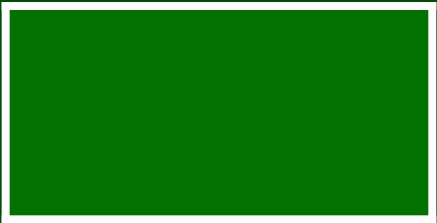


```
load_vector(vector_t *v1);
mul_double(double_t *a);
push();
load_vector(vector_t *v2);
mul_double(double_t *b);
add_double(pop());
```

MPAC



MODE





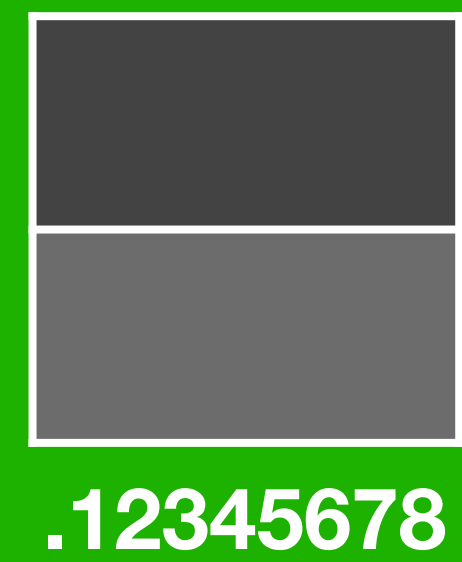
Math Support

Single



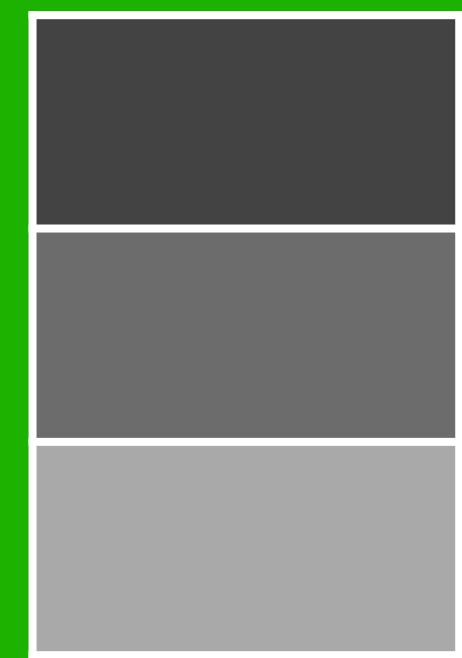
+0

Double



+1

Triple



.12345678012

Vector

-1



Matrix

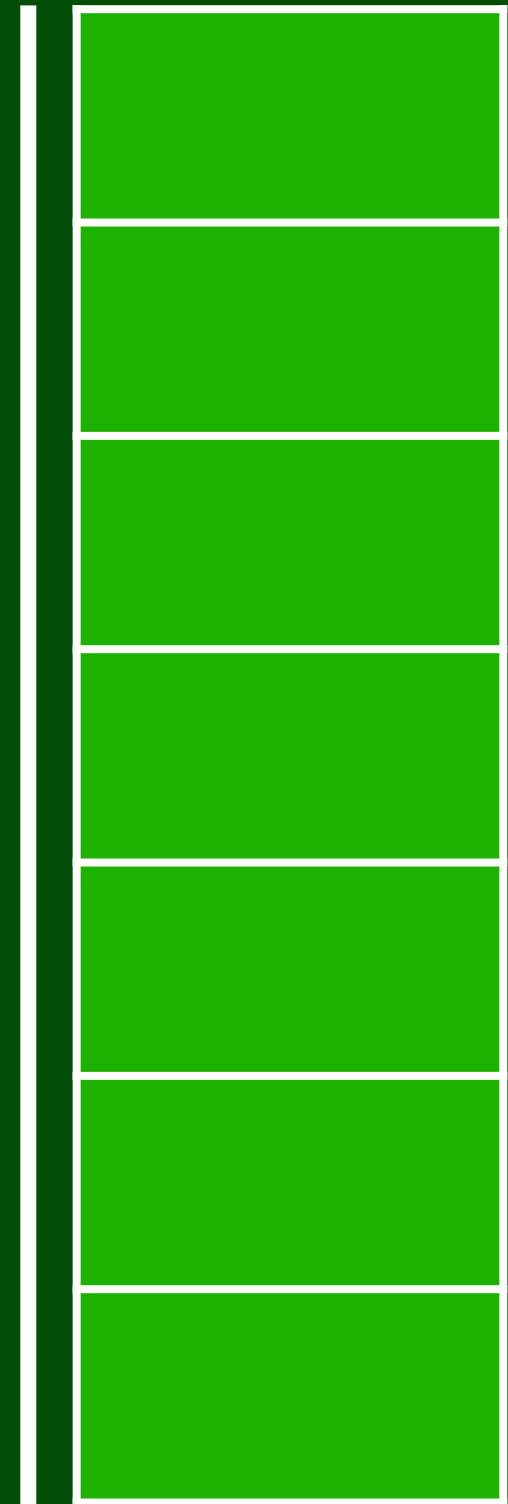


$$a \cdot \vec{v_1} + b \cdot \vec{v_2}$$

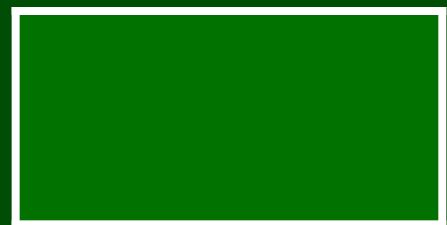


```
load_vector(vector_t *v1);  
mul_double(double_t *a);  
push();  
load_vector(vector_t *v2);  
mul_double(double_t *b);  
add_double(pop());
```

MPAC

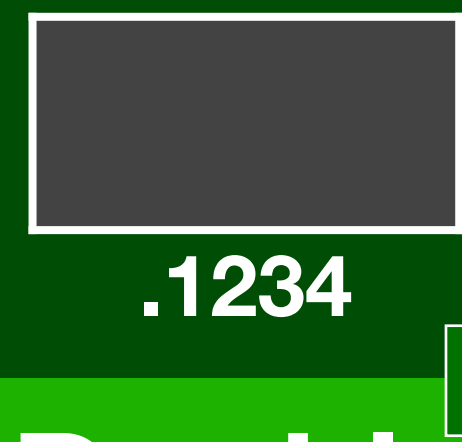


MODE

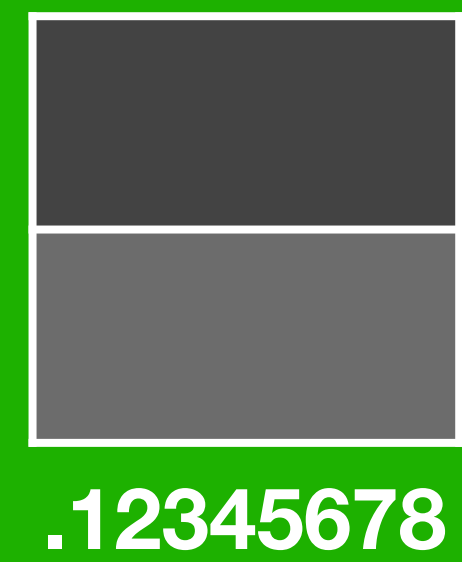


Math Support

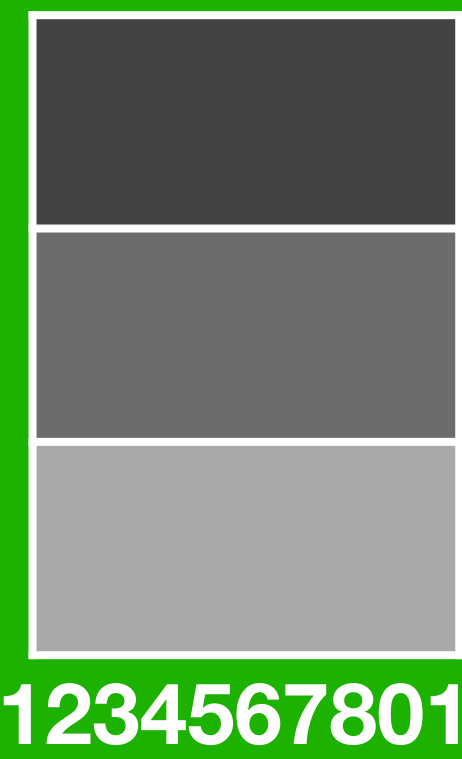
Single



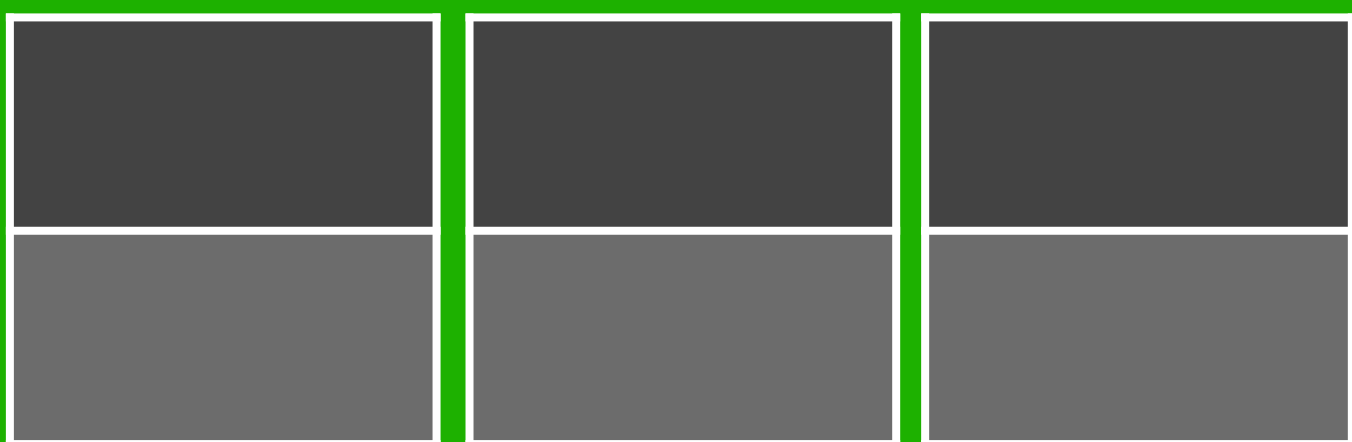
Double



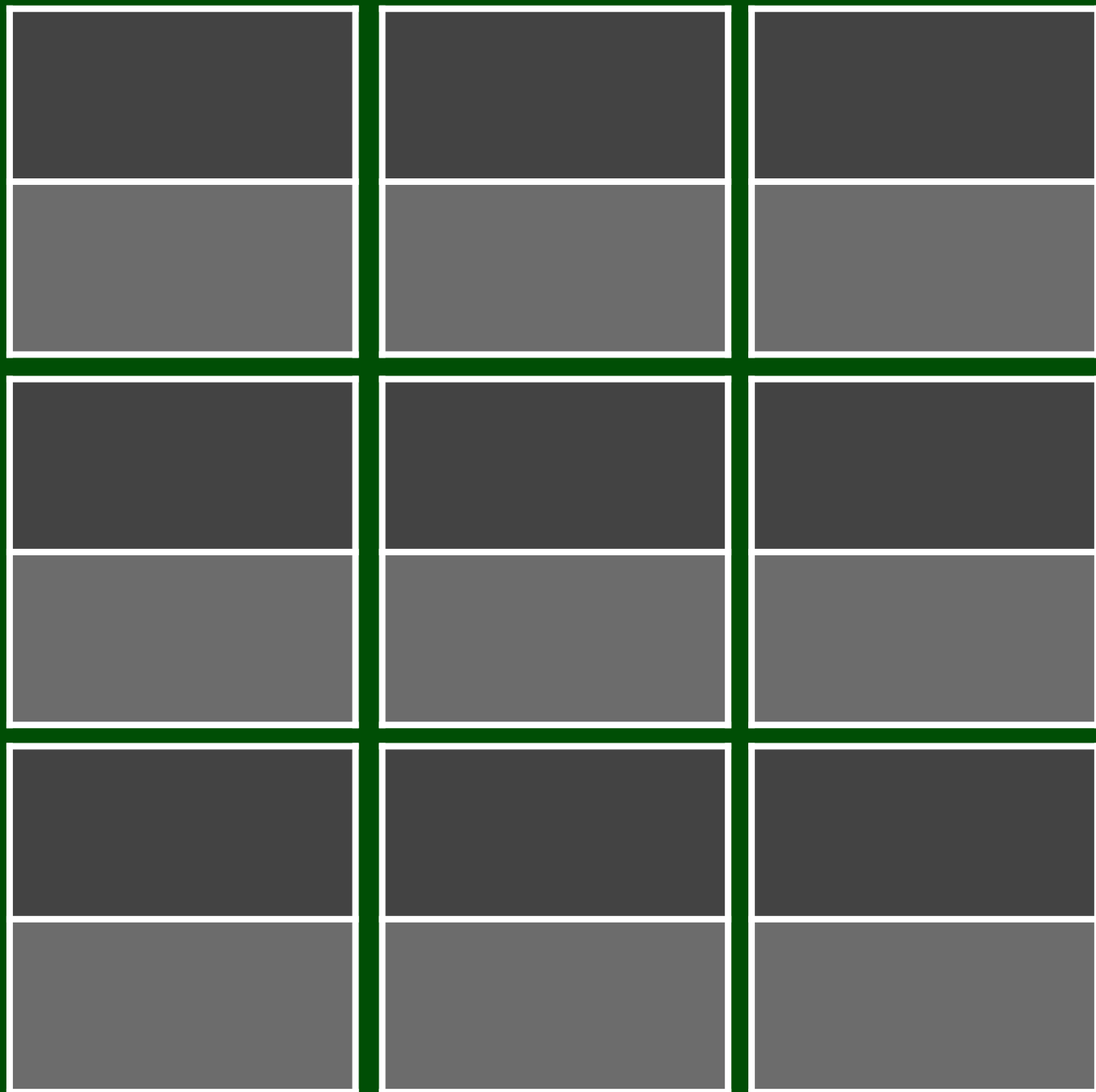
Triple



Vector



Matrix

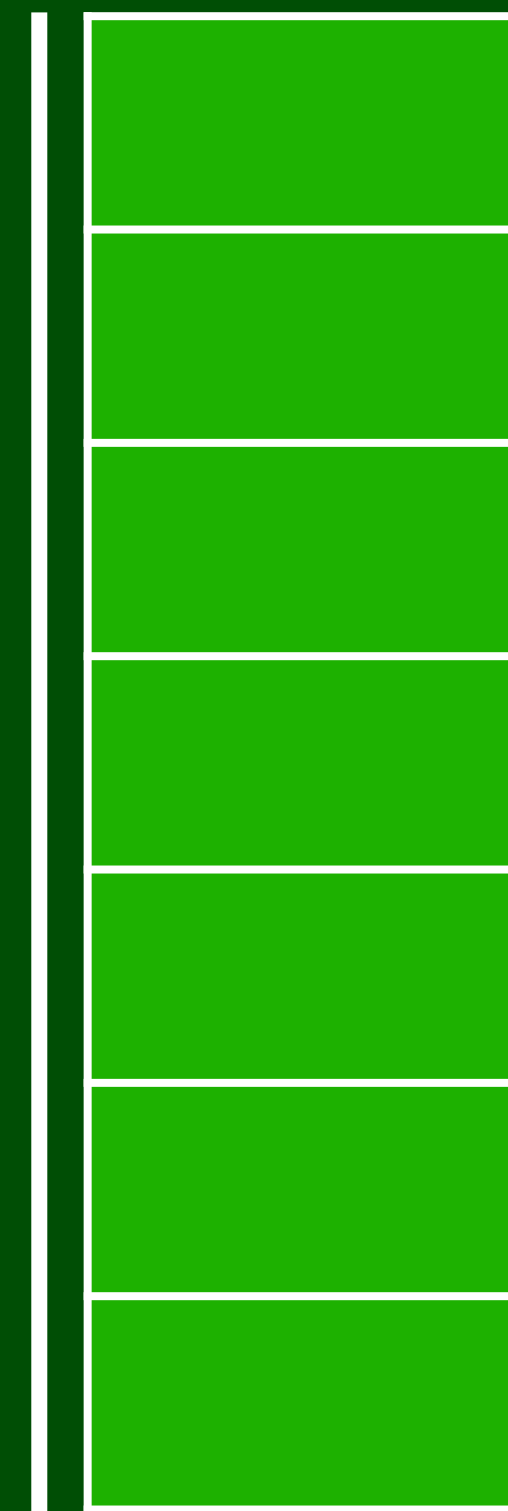


$$a \cdot \vec{v_1} + b \cdot \vec{v_2}$$

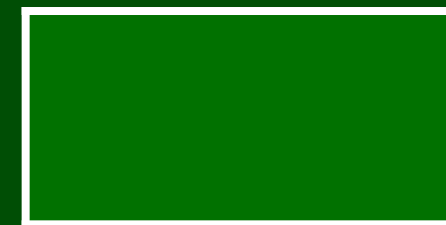


```
load_vector(vector_t *v1);  
mul_double(double_t *a);  
push();  
load_vector(vector_t *v2);  
mul_double(double_t *b);  
add_double(pop());
```

MPAC



MODE



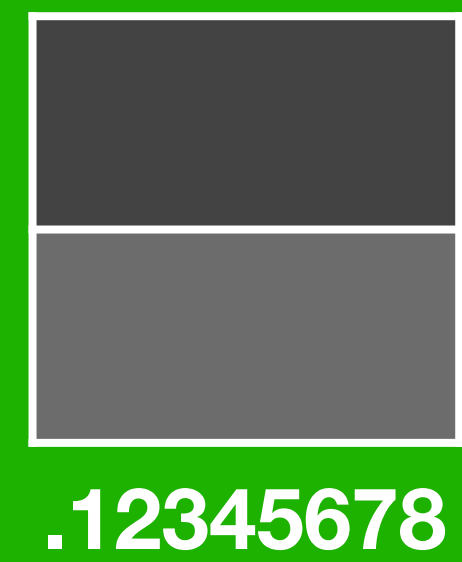
Math Support

Single



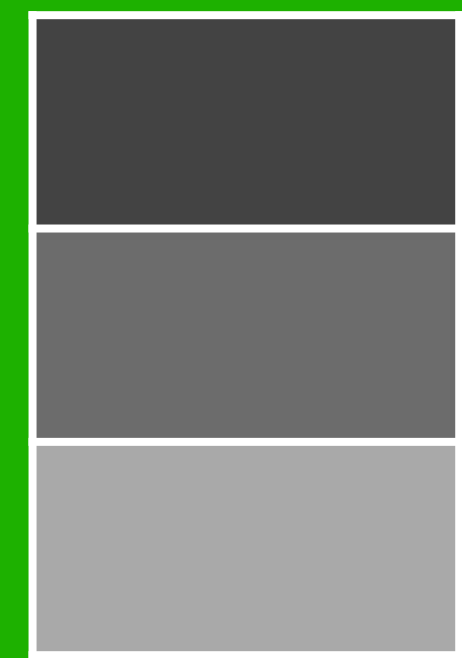
+0

Double



+1

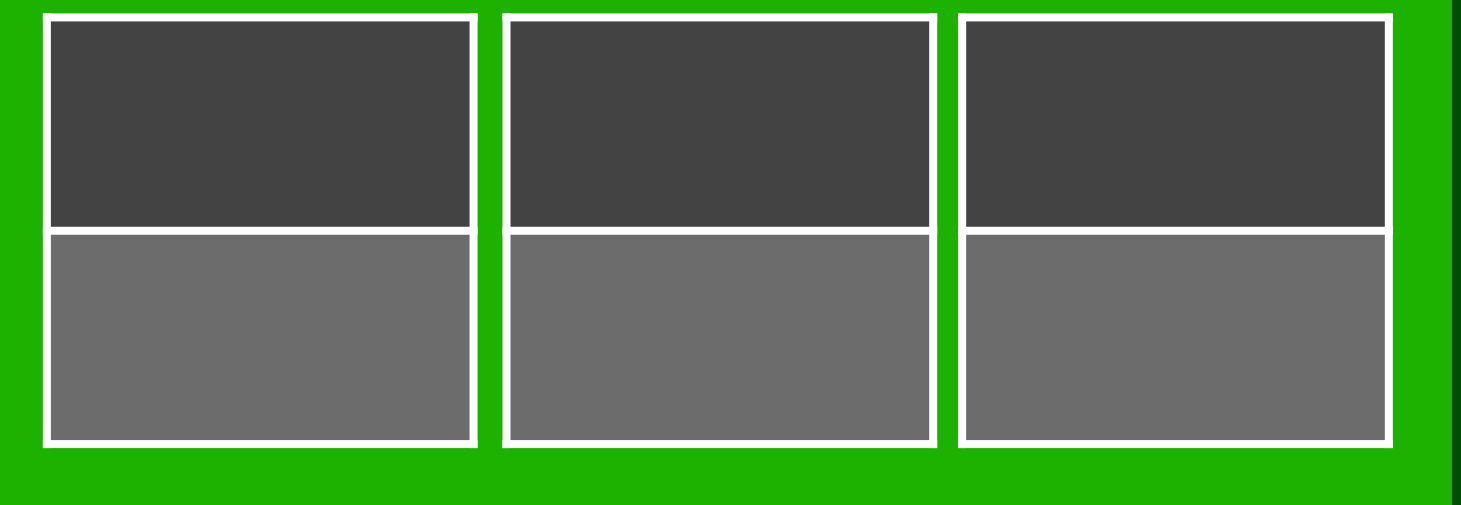
Triple



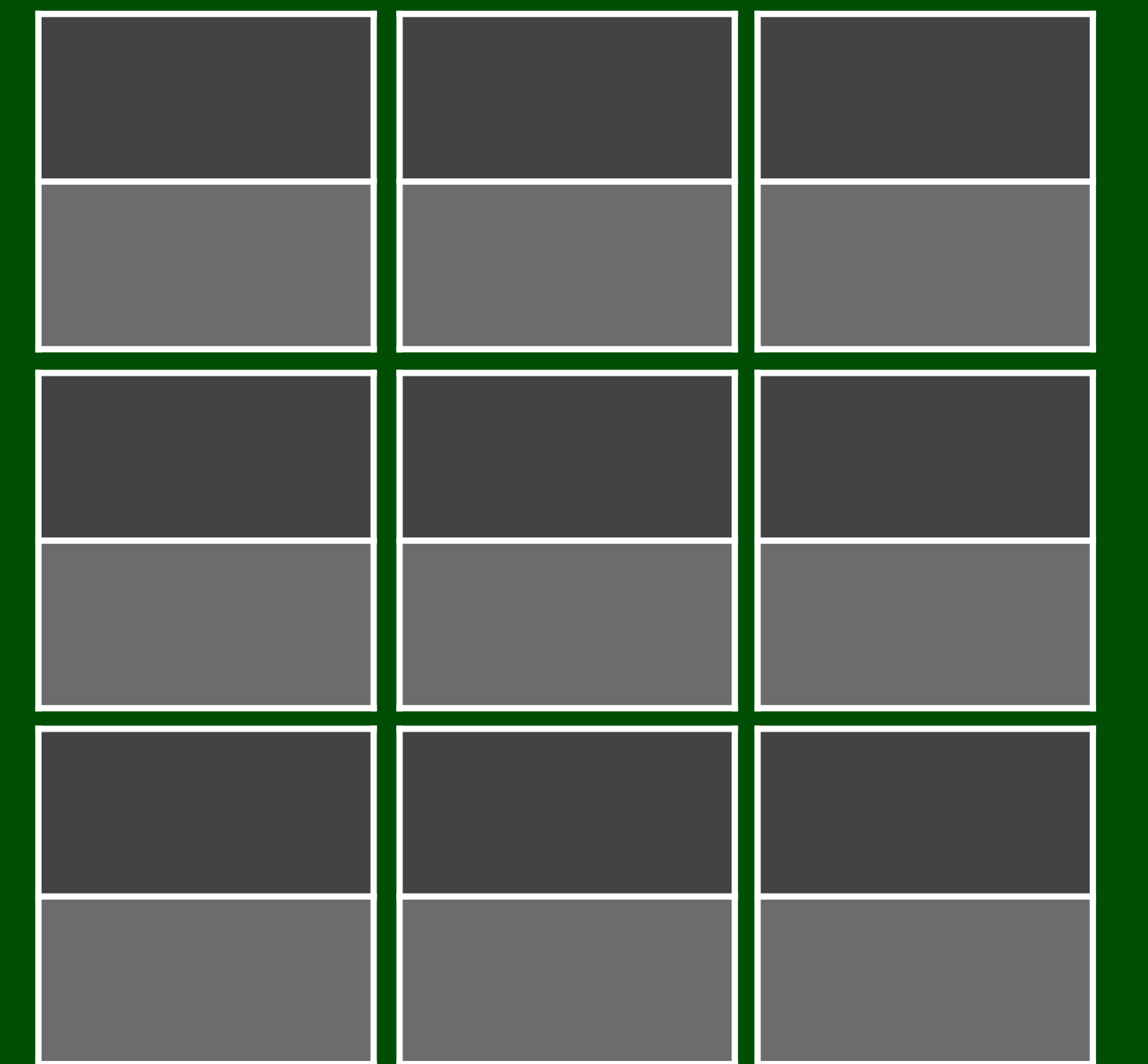
.12345678012

Vector

-1



Matrix

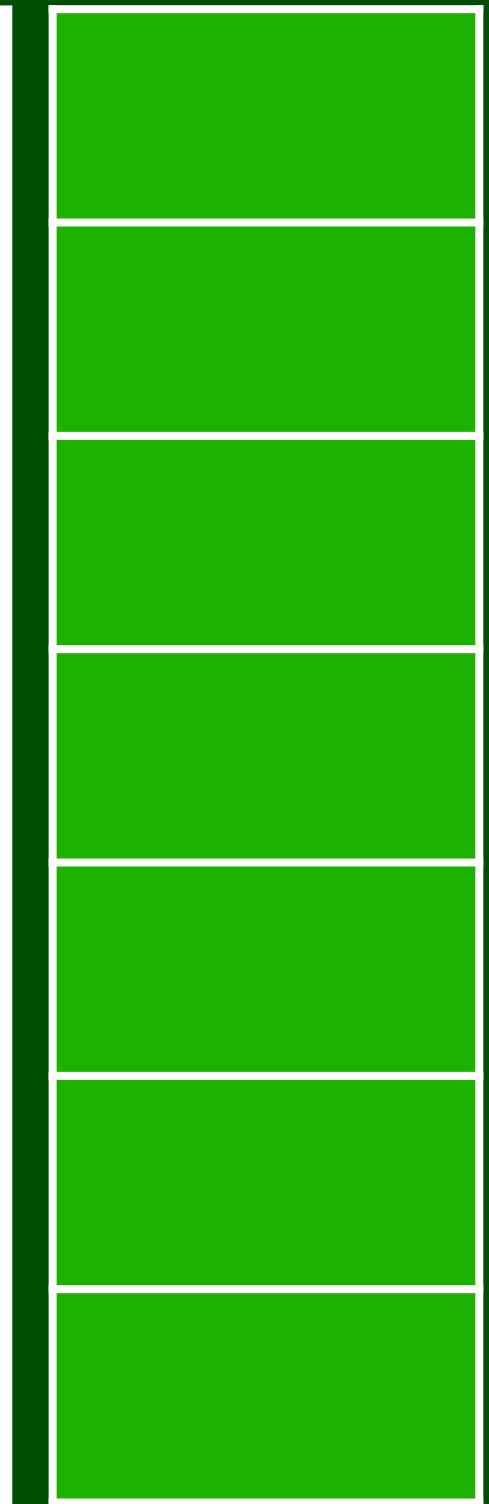


$$a \cdot \vec{v_1} + b \cdot \vec{v_2}$$



```
load_vector(vector_t *v1);
mul_double(double_t *a);
push();
load_vector(vector_t *v2);
mul_double(double_t *b);
add_double(pop());
```

MPAC

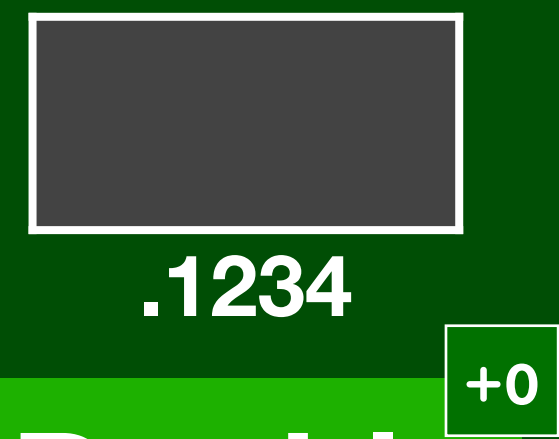


MODE

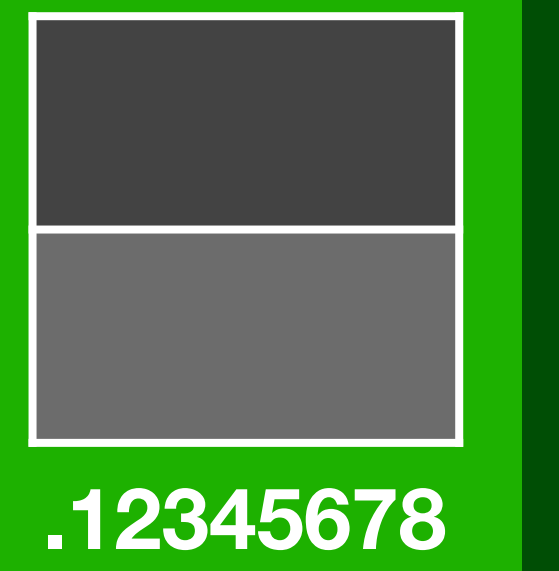


Math Support

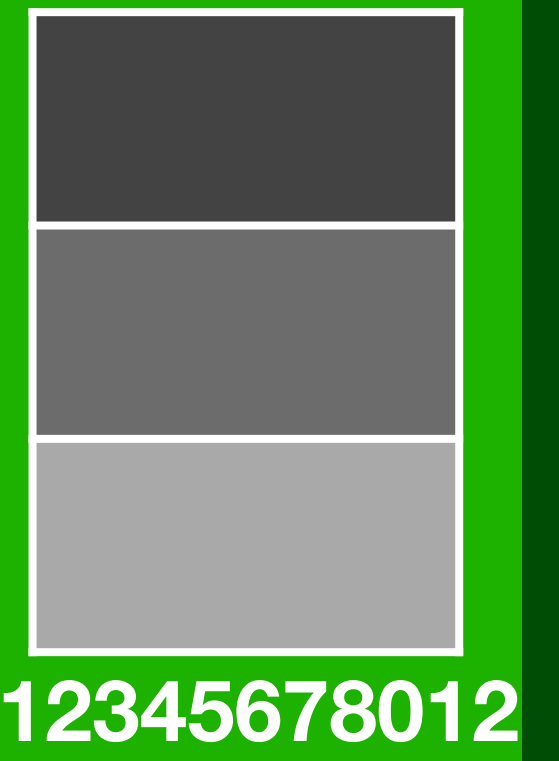
Single



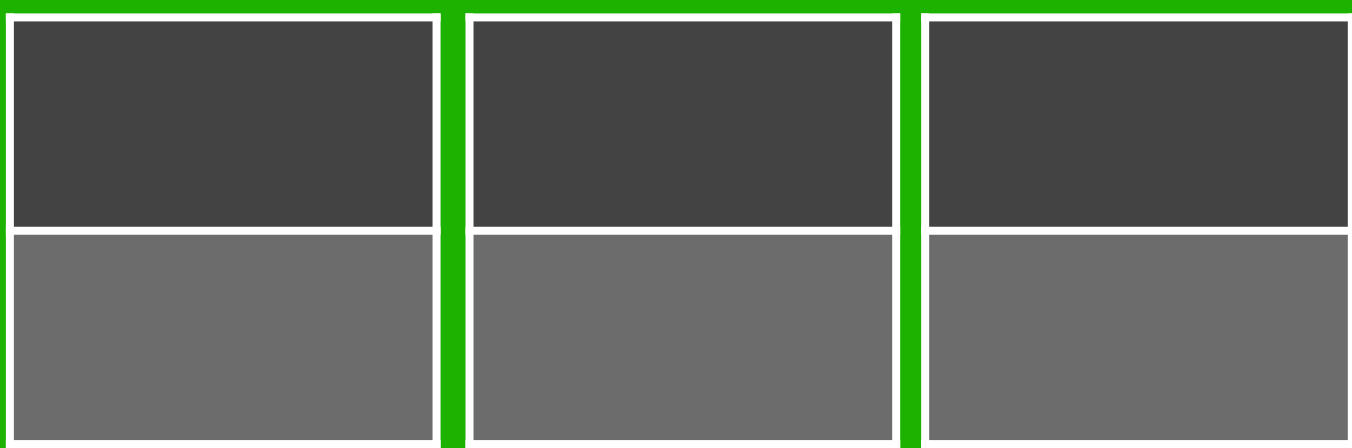
Double



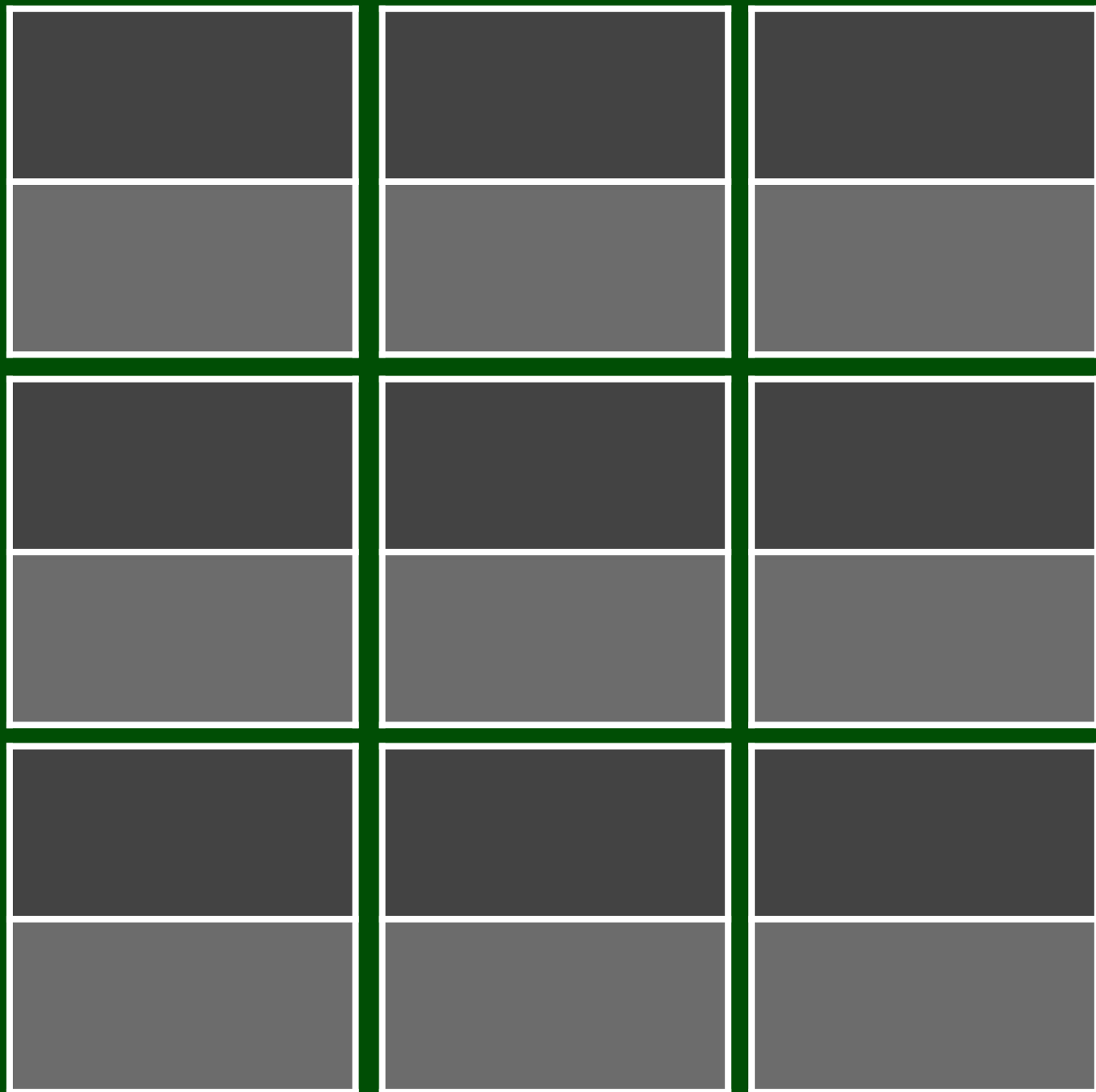
Triple



Vector



Matrix

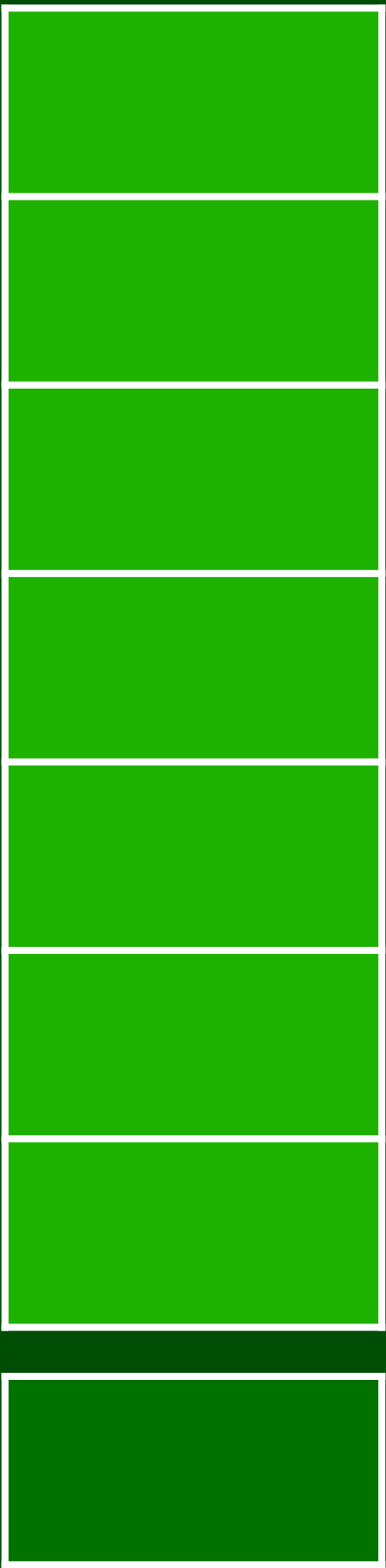


$$a \cdot \vec{v_1} + b \cdot \vec{v_2}$$

```
load_vector(vector_t *v1);  
mul_double(double_t *a);  
push();  
load_vector(vector_t *v2);  
mul_double(double_t *b);  
add_double(pop());
```

MPAC

MODE





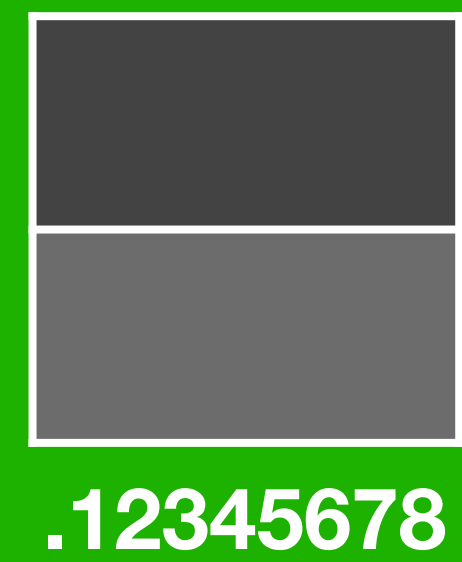
Math Support

Single



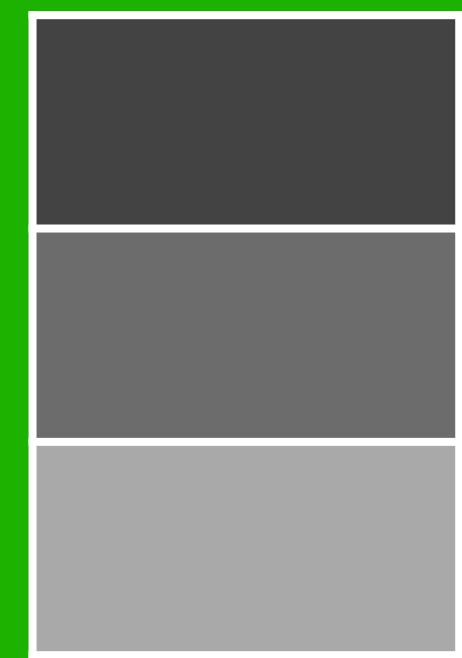
+0

Double



+1

Triple



.12345678012

Vector

-1



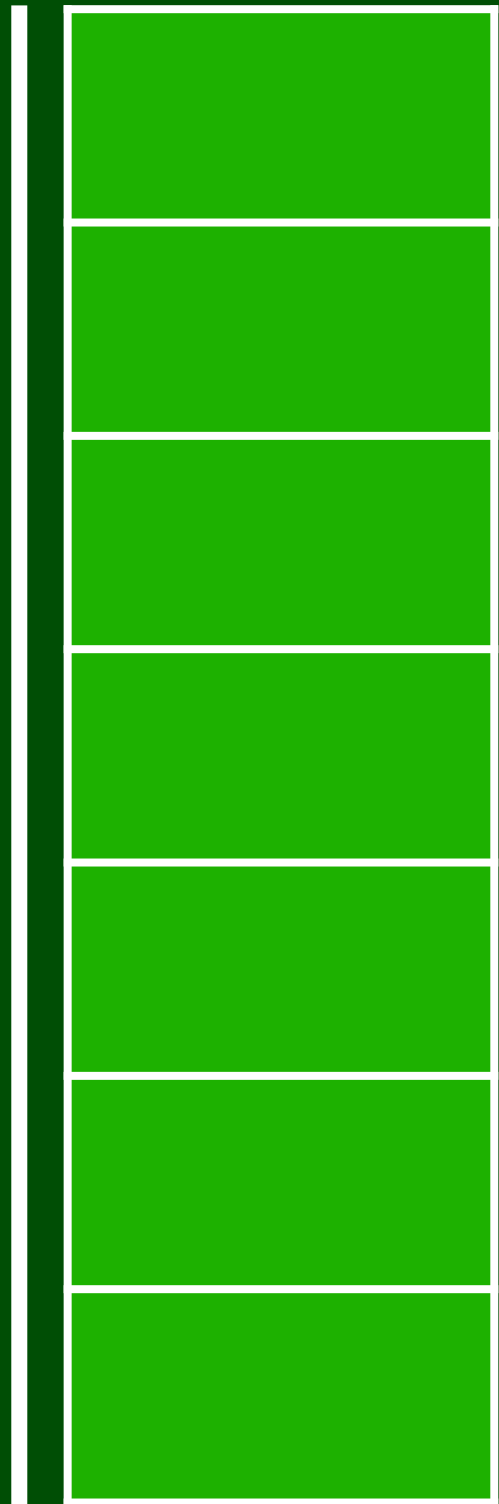
Matrix



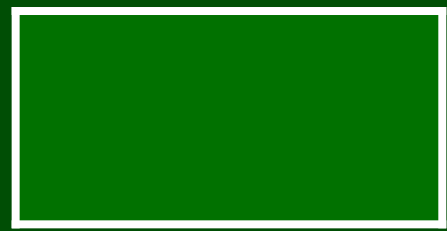
$$a \cdot \vec{v_1} + b \cdot \vec{v_2}$$

```
load_vector(vector_t *v1);  
mul_double(double_t *a);  
push();  
load_vector(vector_t *v2);  
mul_double(double_t *b);  
add_double(pop());
```

MPAC

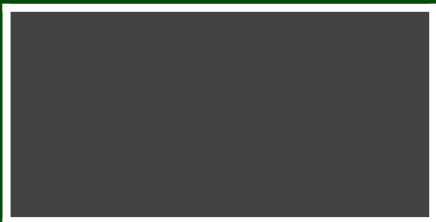


MODE



Math Support

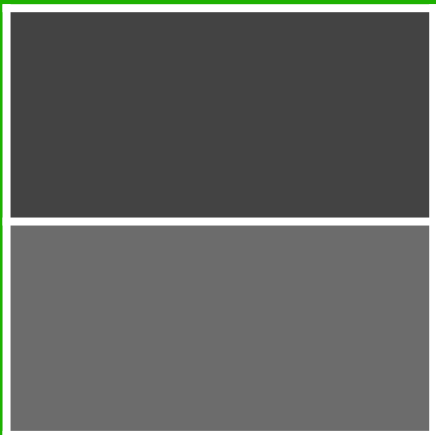
Single



.1234

+0

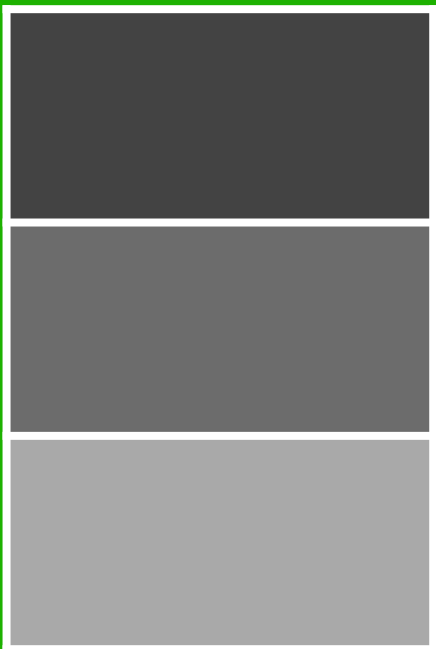
Double



.12345678

+1

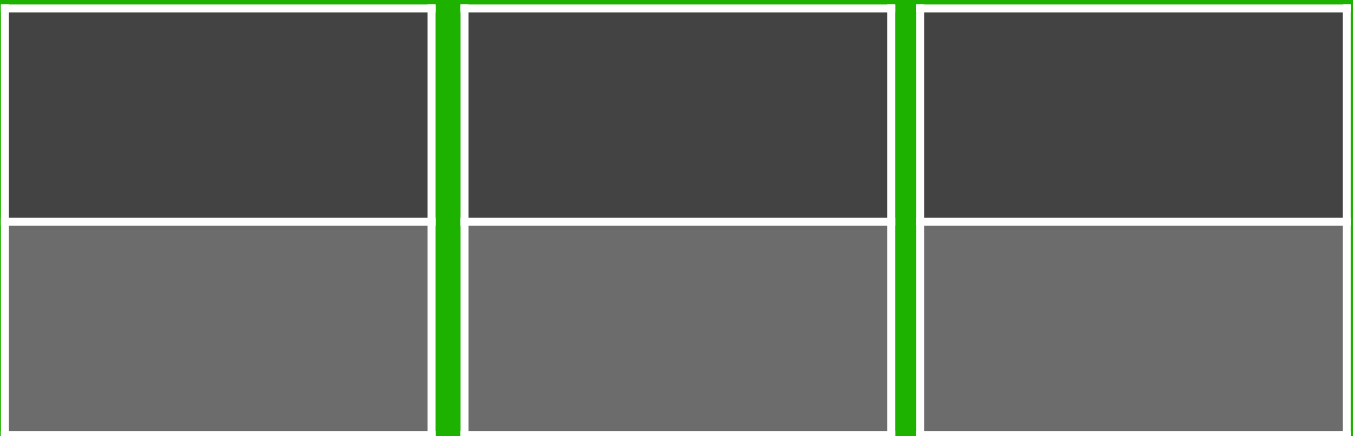
Triple



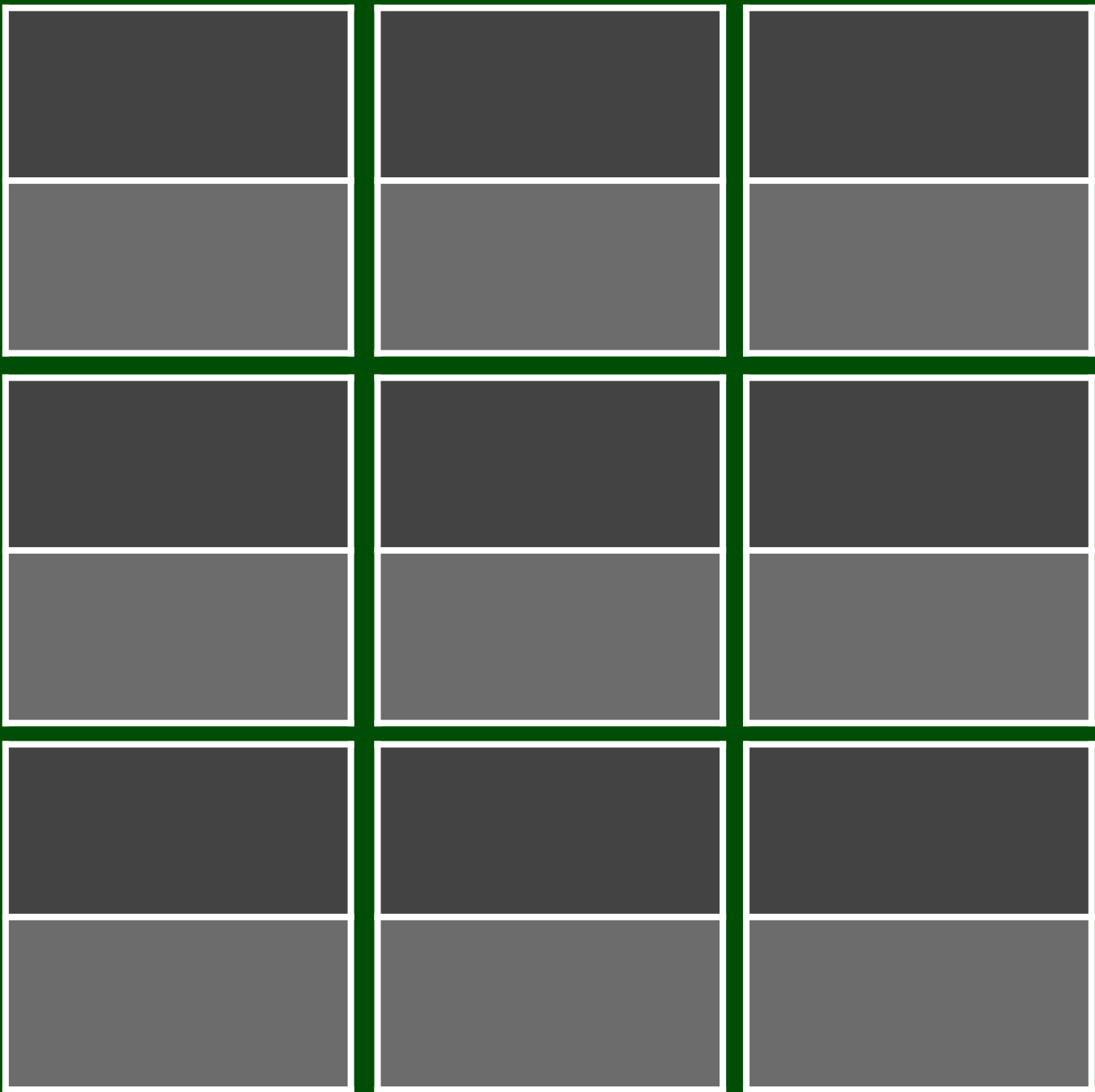
.12345678012

Vector

-1



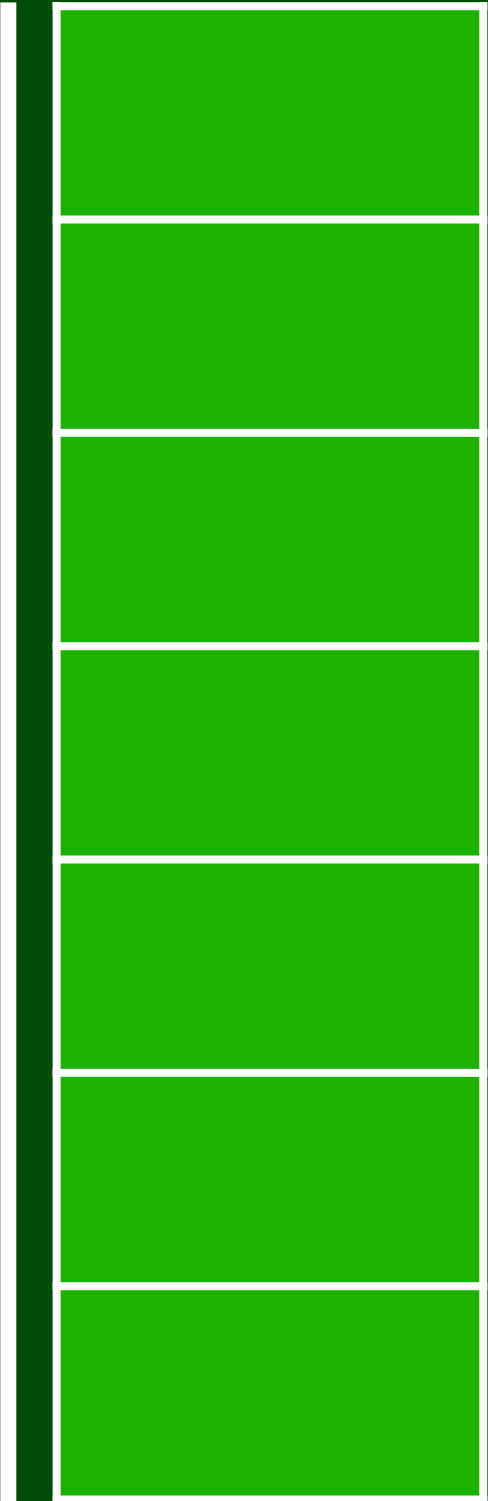
Matrix



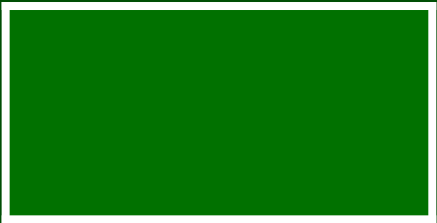
$$a \cdot \vec{v_1} + b \cdot \vec{v_2}$$

```
load_vector(vector_t *v1);  
mul_double(double_t *a);  
push();  
load_vector(vector_t *v2);  
mul_double(double_t *b);  
add_double(pop());
```

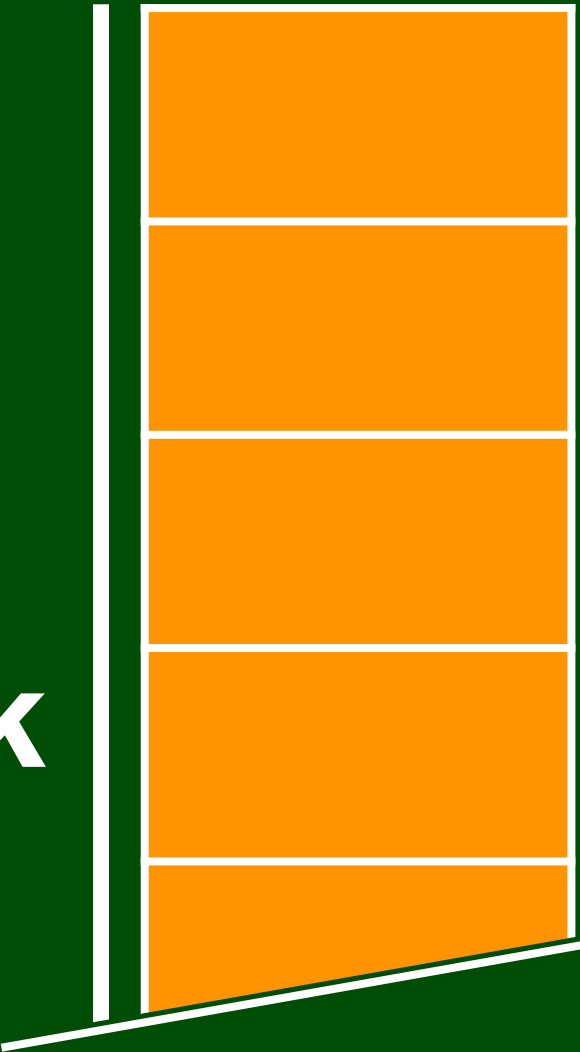
MPAC



MODE



Stack  
(38 words)



Stack Ptr



VAC

## Core Set Entry

MPAC

MODE

PC

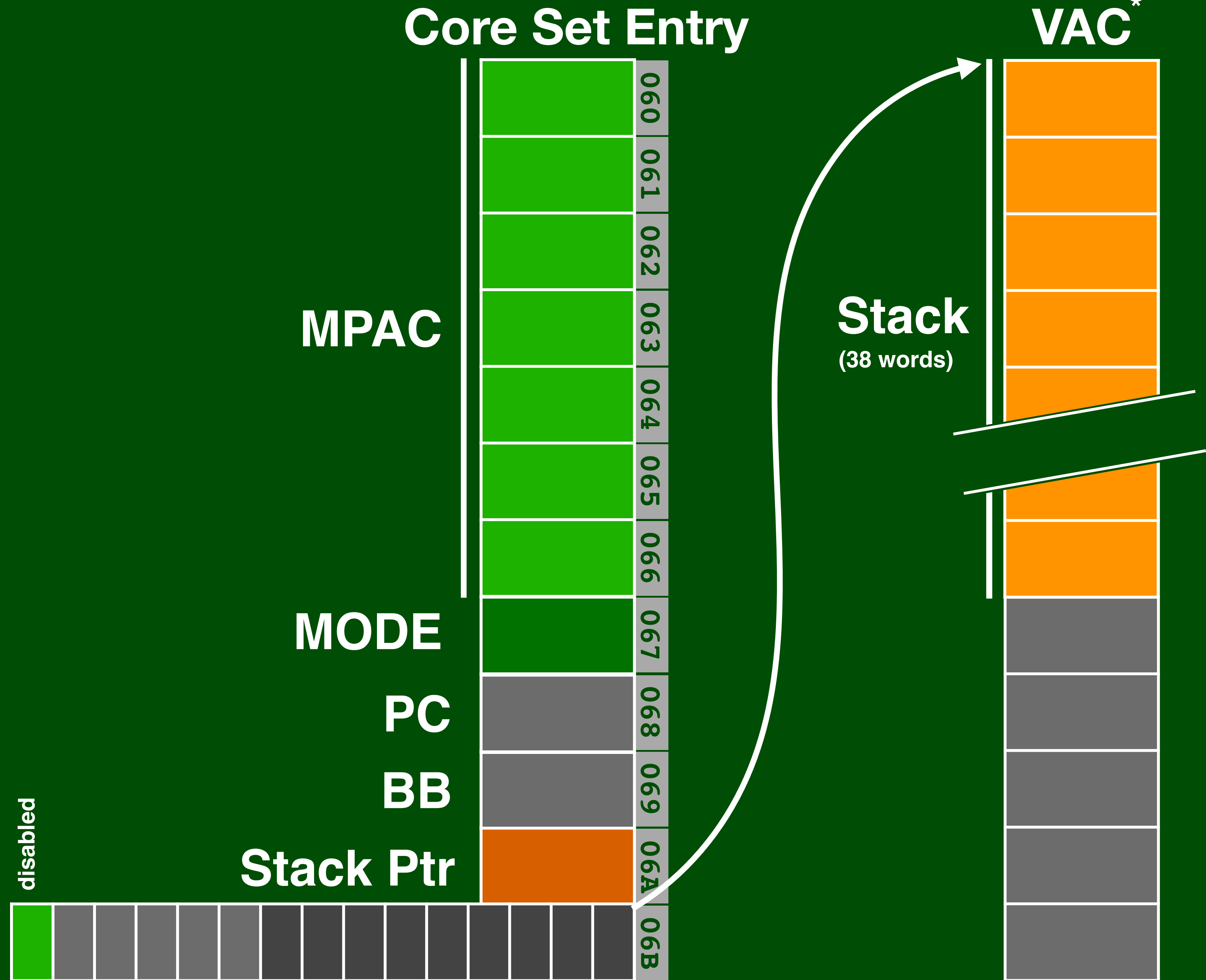
BB

Stack Ptr

Priority

060  
061  
062  
063  
064  
065  
066  
067  
068  
069  
06A  
06B

Stack  
(38 words)



\* VAC = Vector Accumulator, a misnomer



VAC

**NOVAC**

Create new job

**FINDVAC**

Create new job + VAC

**MPAC**

**MODE**

**PC**

**BB**

**Stack Ptr**

disabled

priority

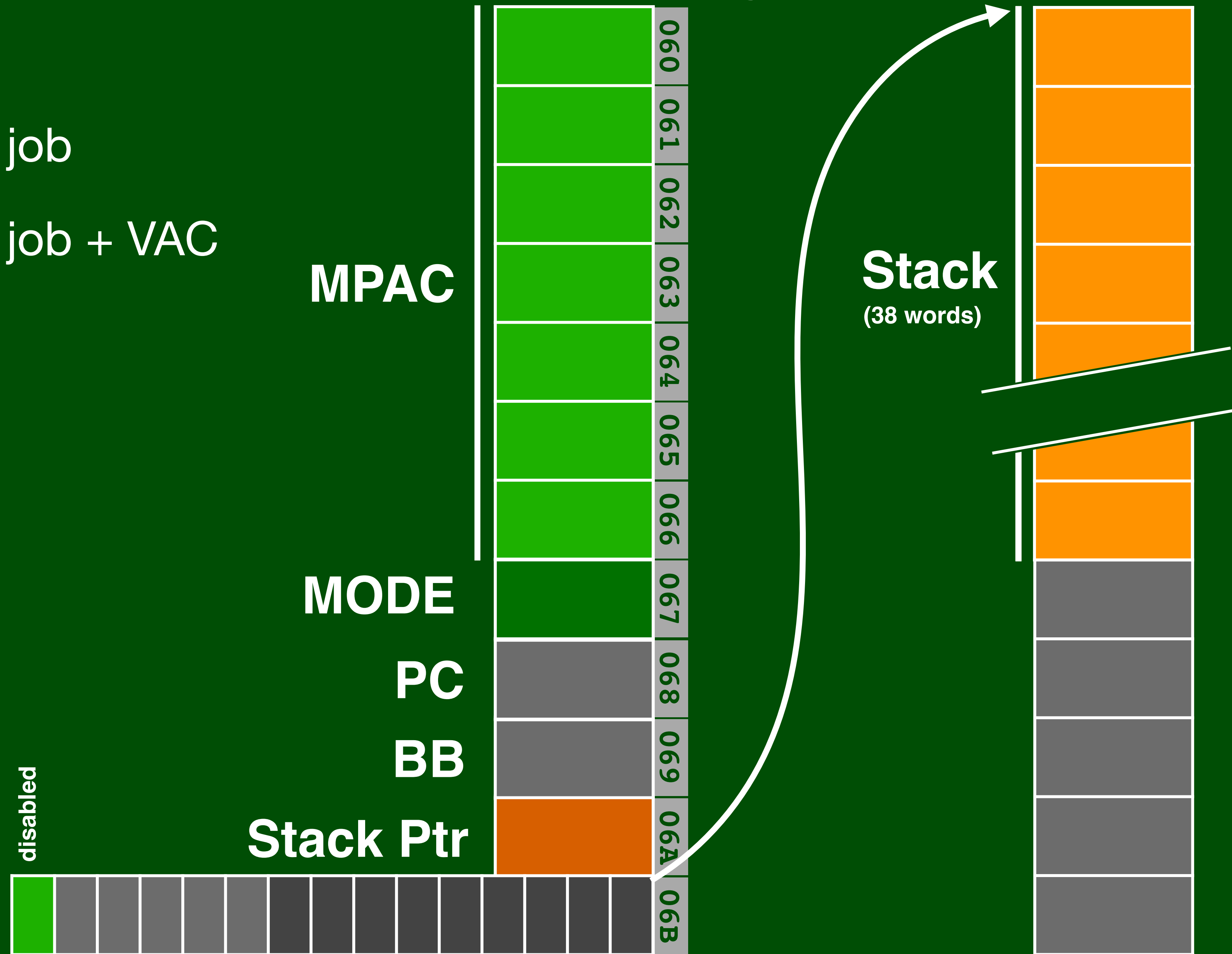
VAC pointer

**Core Set Entry**

**VAC\***

**Stack**  
(38 words)

\* VAC = Vector Accumulator, a misnomer



VAC

**NOVAC**

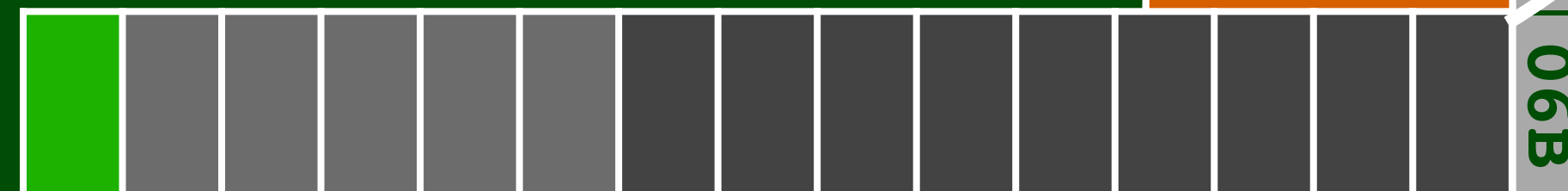
Create new job

**FINDVAC**

Create new job + VAC

```
load_vector(vector_t *v1);  
mul_double(double_t *a);  
push();  
load_vector(vector_t *v2);  
mul_double(double_t *b);  
add_double(pop());
```

disabled



priority

VAC pointer

Core Set Entry

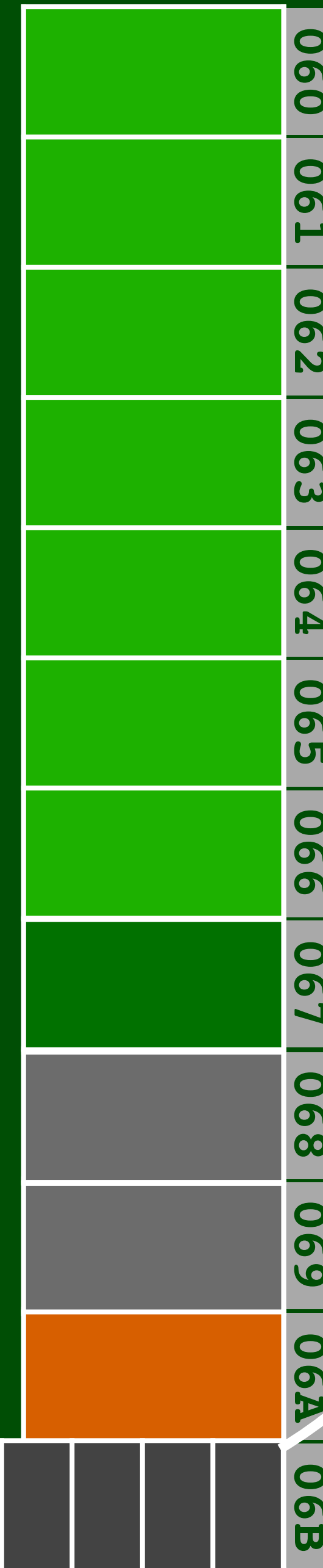
MPAC

MODE

PC

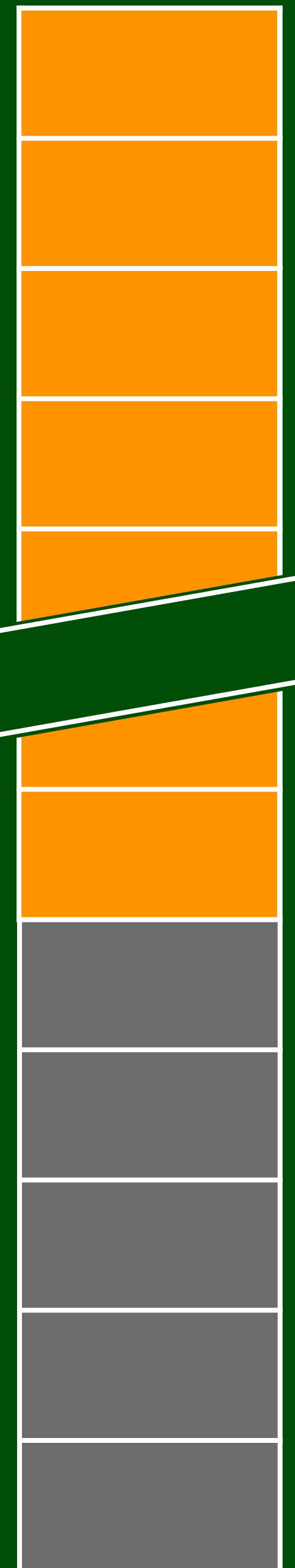
BB

Stack Ptr



Stack  
(38 words)

VAC\*



\* VAC = Vector Accumulator, a misnomer

VAC

**NOVAC**

Create new job

**FINDVAC**

Create new job + VAC

**VLOAD V1**

**DMP A**

**PUSH**

**VLOAD V2**

**DMP B**

**DAD**

Core Set Entry

MPAC

MODE

PC

BB

Stack Ptr

disabled

priority

VAC pointer

Stack  
(38 words)

VAC\*

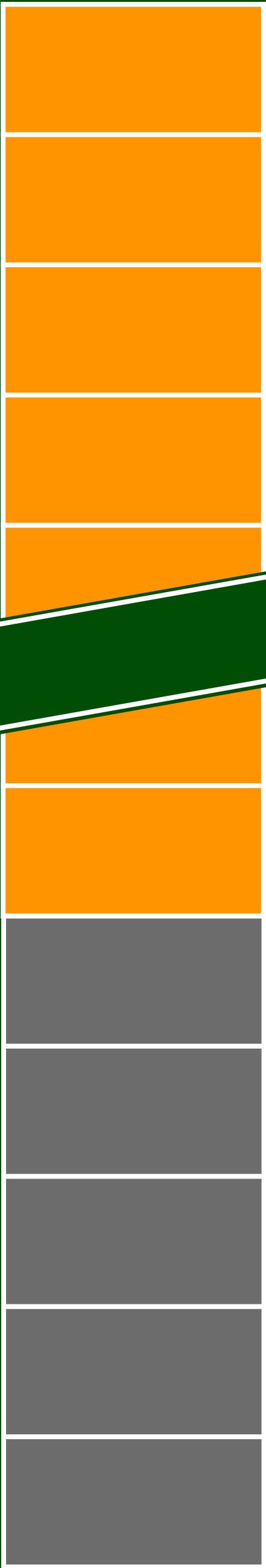
\* VAC = Vector Accumulator, a misnomer

**Interpretive Code**

**VLOAD V1**  
**DMP A**  
**PUSH**  
**VLOAD V2**  
**DMP B**  
**DAD**

**Stack**  
(38 words)

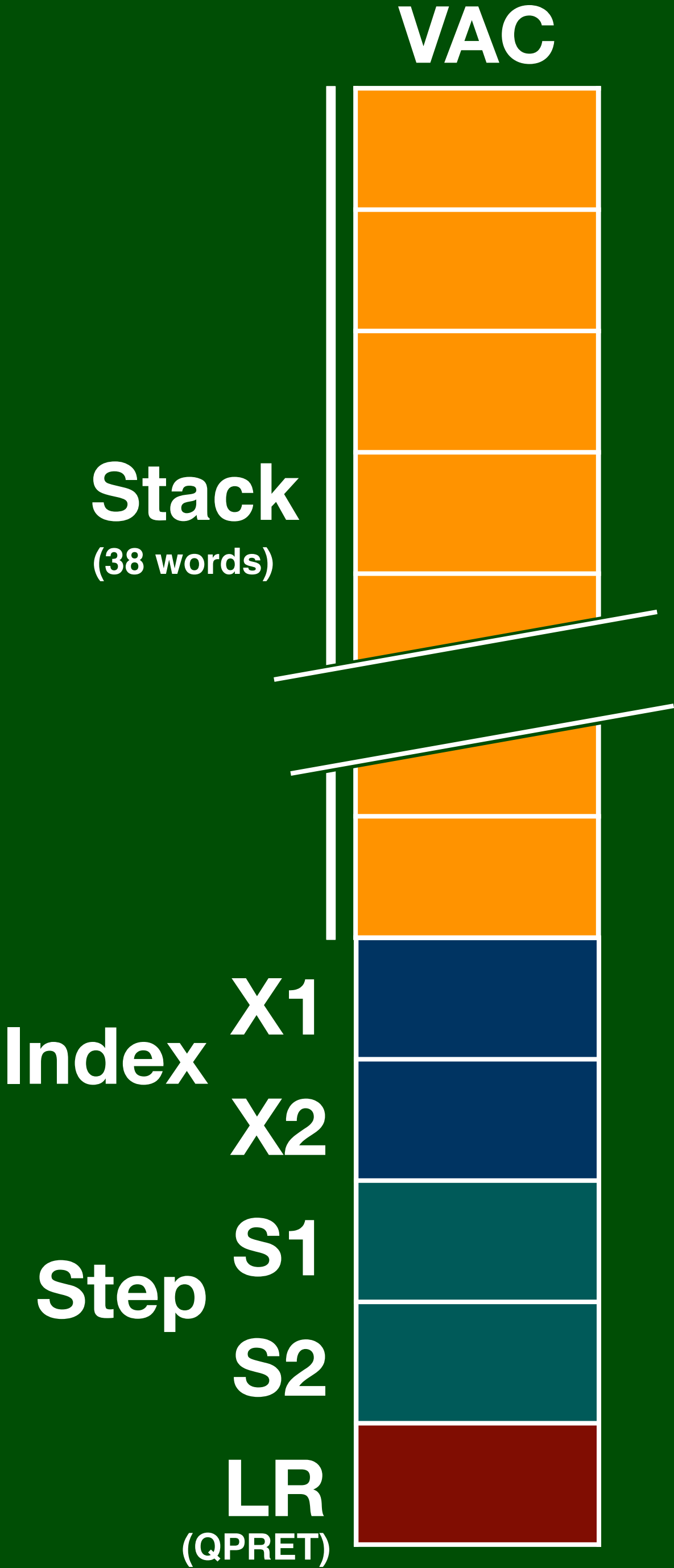
**VAC**





Interpretive Code

VLOAD V1  
DMP A  
PUSH  
VLOAD V2  
DMP B  
DAD



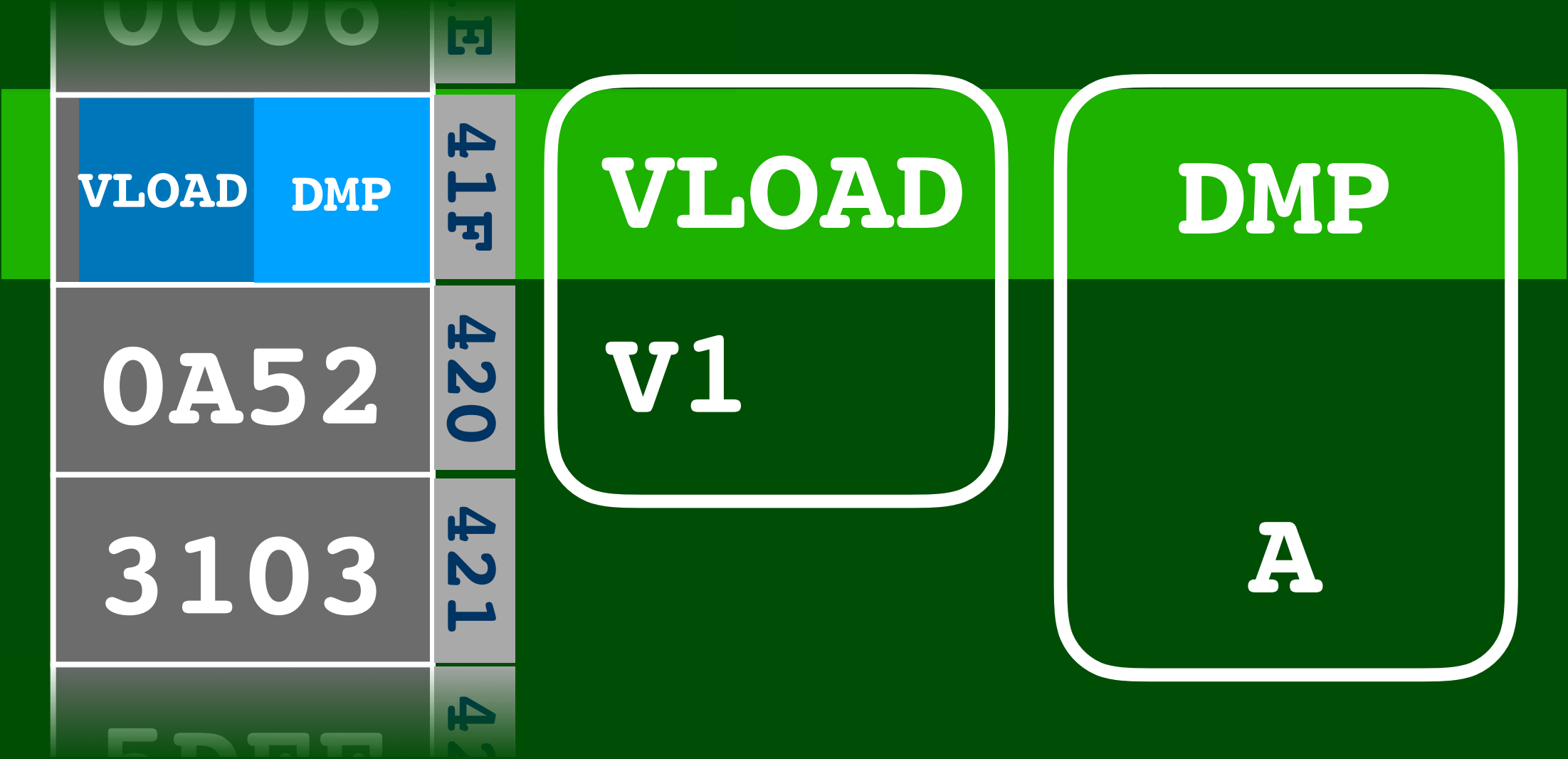
# Interpreter Encoding

0000	E
VLOAD DMP	41F
0A52	420
3103	421
5DEF	422

VLOAD  
v1

DMP  
A

# Interpreter Encoding



Interpreter Encoding

128 opcodes

Opcode 1  
(7 bit)

Opcode 2  
(7 bit)

0000	41F
VLOAD DMP	41F
0A52	420
3103	421
5DEF	422

VLOAD

v1

DMP

A

SHR7

0000

013



Interpreter Encoding

128 opcodes

Opcode 1  
(7 bit)

Opcode 2  
(7 bit)

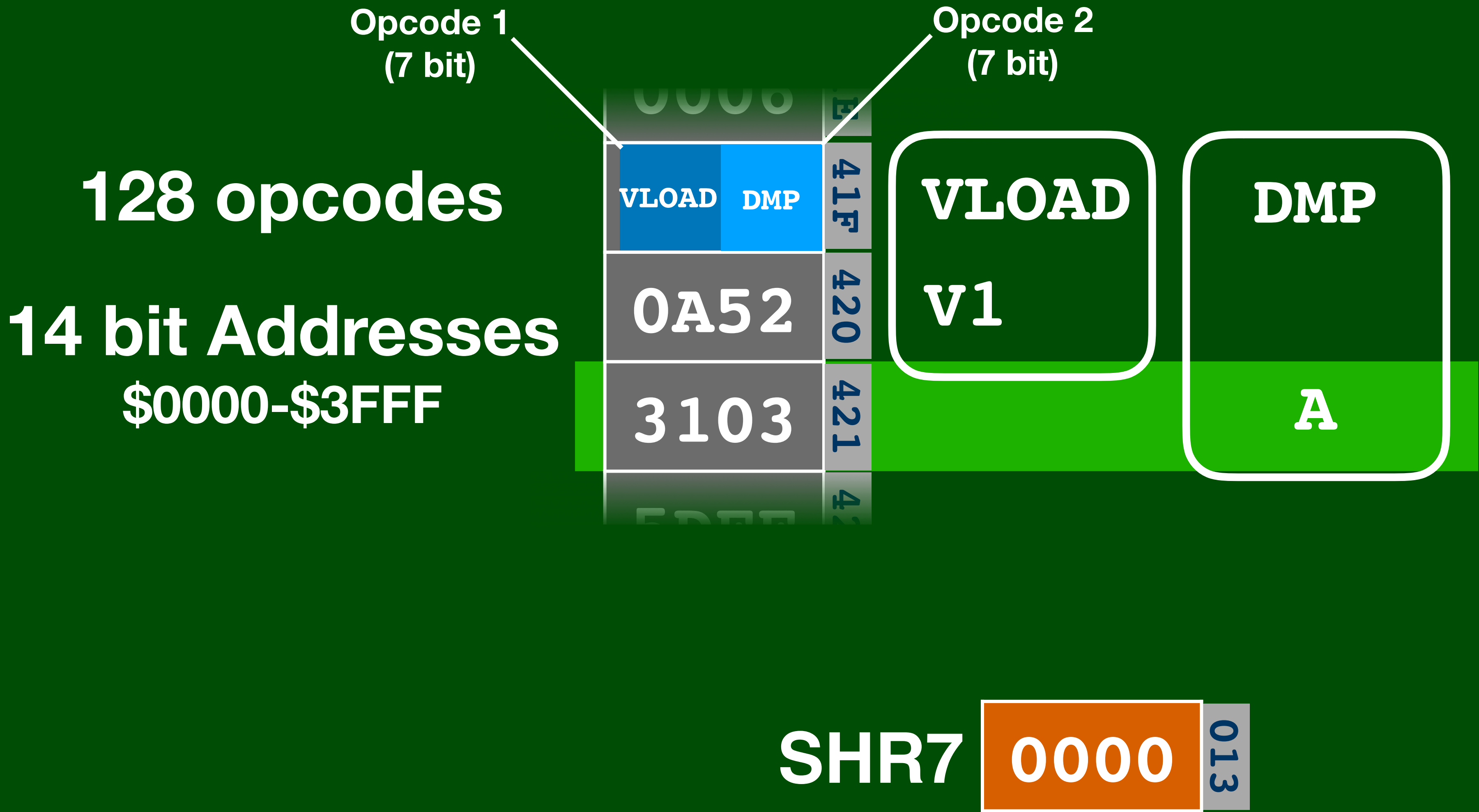


SHR7

0000

013

Interpreter Encoding

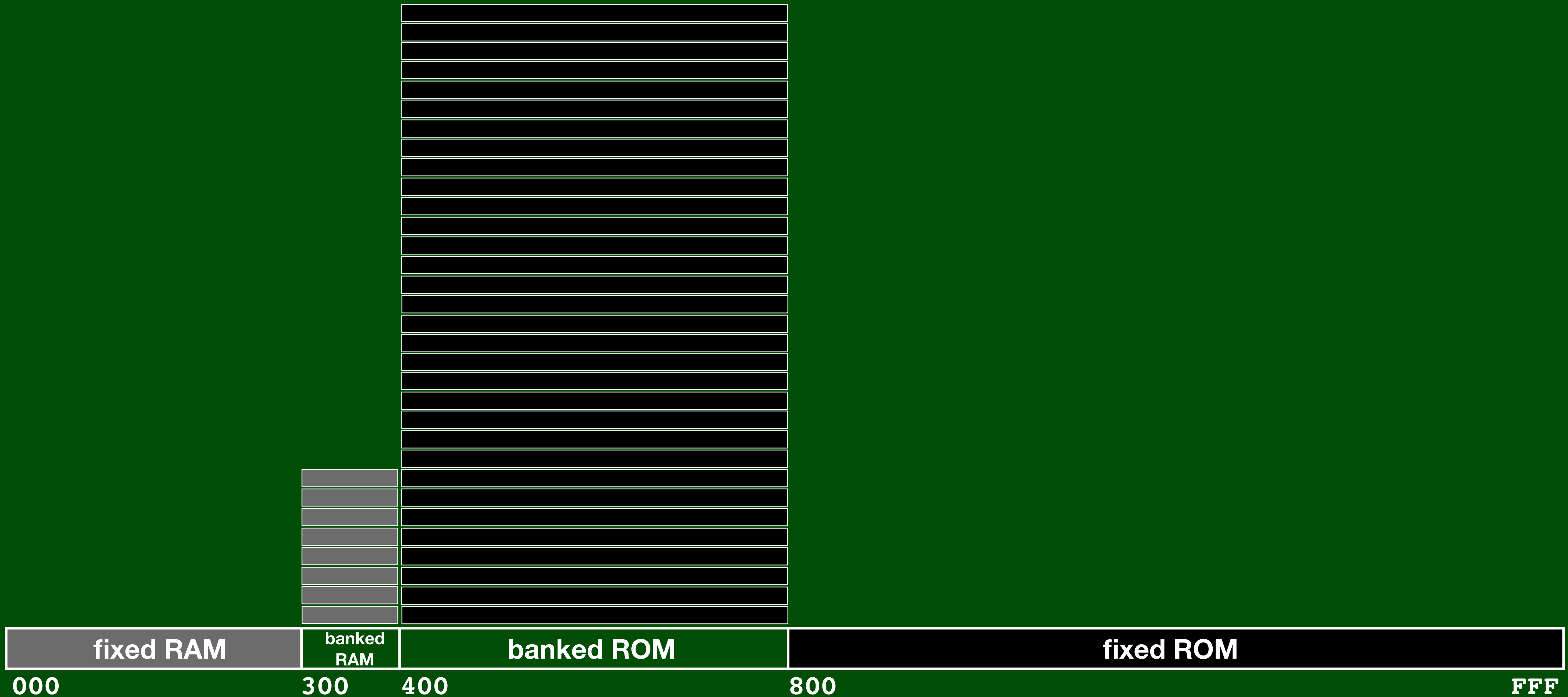


Interpreter Memory

**14 bit Addresses**  
**\$0000-\$3FFF**

Interpreter Memory

14 bit Addresses  
\$0000-\$3FFF





# Interpreter Memory

**14 bit Addresses**  
**\$0000-\$3FFF**



# Interpreter Memory

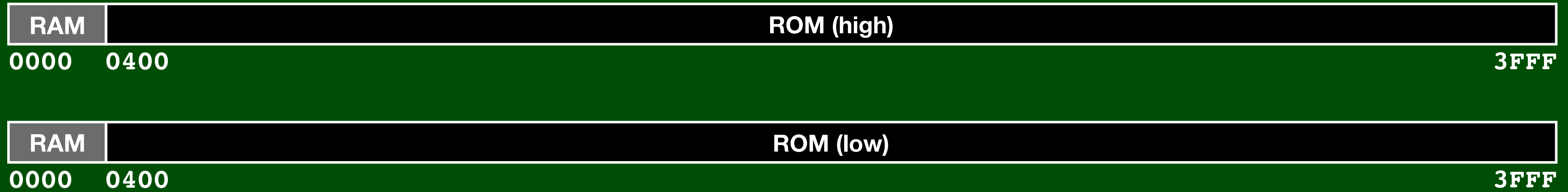
**14 bit Addresses**  
**\$0000-\$3FFF**



# Interpreter Memory



# Interpreter Memory





# Interpreter Instructions

1. **Instruction** (Instruction)

2. **Instruction** (Instruction)

3. **Instruction** (Instruction)

4. **Instruction** (Instruction)

5. **Instruction** (Instruction)

6. **Instruction** (Instruction)

7. **Instruction** (Instruction)

8. **Instruction** (Instruction)

9. **Instruction** (Instruction)

10. **Instruction** (Instruction)

11. **Instruction** (Instruction)

12. **Instruction** (Instruction)

13. **Instruction** (Instruction)

14. **Instruction** (Instruction)

# Interpreter Instructions

## Load/Store

<b>SLOAD</b>
<b>DLOAD</b>
<b>TLOAD</b>
<b>VLOAD</b>
<b>STORE</b>
<b>STODL</b>
<b>STOVL</b>
<b>STCALL</b>

## Arithmetic

<b>DAD</b>
<b>DSU</b>
<b>BDSU</b>
<b>DMP</b>
<b>DMPR</b>
<b>DDV</b>
<b>BDDV</b>
<b>SIGN</b>
<b>TAD</b>

<b>SQRT</b>
<b>SIN</b>
<b>COS</b>
<b>ARCSIN</b>
<b>ARCCOS</b>
<b>DSQ</b>
<b>ROUND</b>
<b>DCOMP</b>
<b>ABS</b>

## Vector

<b>VAD</b>
<b>VSU</b>
<b>BVSU</b>
<b>DOT</b>
<b>VXSC</b>
<b>V/SC</b>
<b>VXV</b>
<b>VPROJ</b>
<b>VXM</b>
<b>MXV</b>

## Shift

<b>SR</b>
<b>SRR</b>
<b>SL</b>
<b>SLR</b>
<b>VSR</b>
<b>VSL</b>

## Index

<b>AXT</b>
<b>AXC</b>
<b>LXA</b>
<b>LXC</b>
<b>SXA</b>
<b>XCHX</b>
<b>SSP</b>
<b>INCR</b>
<b>XAD</b>
<b>XSU</b>
<b>TIX</b>

## Control Flow

<b>GOTO</b>
<b>CALL</b>
<b>RVQ</b>
<b>STQ</b>
<b>CGOTO</b>
<b>CCALL</b>
<b>BPL</b>
<b>BZE</b>
<b>BMN</b>
<b>BHIZ</b>
<b>EXIT</b>
<b>RTB</b>
<b>BOV</b>
<b>BOVB</b>

## Stack

<b>PUSH</b>
<b>PDDL</b>
<b>PDVL</b>

# Interpreter Instructions

## Load/Store

SLOAD
DLOAD
TLOAD
VLOAD
STORE
STODL
STOVL
STCALL

## Arithmetic

DAD
DSU
BDSU
DMP
DMPR
DDV
BDDV
SIGN
TAD

## Vector

SQRT
SIN
COS
ARCSIN
ARCCOS
DSQ
ROUND
DCOMP
ABS

## Shift

SR
SRR
SL
SLR
VSR
VSL

## Index

AXT
AXC
LXA
LXC
SXA
XCHX
SSP
INCR
XAD
XSU
TIX

## Control Flow

GOTO
CALL
RVQ
STQ
CGOTO
CCALL
BPL
BZE
BMN
BHIZ
EXIT
RTB
BOV
BOVB

## Stack

PUSH
PDDL
PDVL

# Interpreter Instructions

## Load/Store

SLOAD
DLOAD
TLOAD
VLOAD
STORE
STODL
STOVL
STCALL

## Arithmetic

DAD
DSU
BDSU
DMP
DMPR
DDV
BDDV
SIGN
TAD

## Vector

SQRT
SIN
COS
ARCSIN
ARCCOS
DSQ
ROUND
DCOMP
ABS

## Shift

VAD
VSU
BVSU
DOT
VXSC
V/SC
VXV
VPROJ
VXM
MXV

## Index

SR
SRR
SL
SLR
VSR
VSL

## Control Flow

AXT
AXC
LXA
LXC
SXA
XCHX
SSP
INCR
XAD
XSU
TIX

## Stack

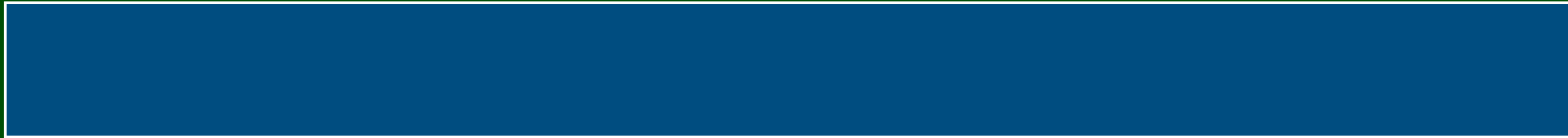
PUSH
PDDL
PDVL

↔ CALL INTPRET

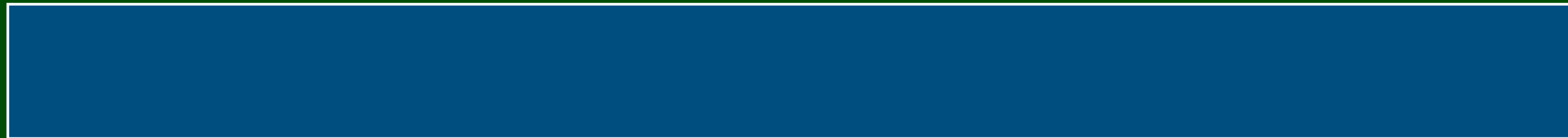
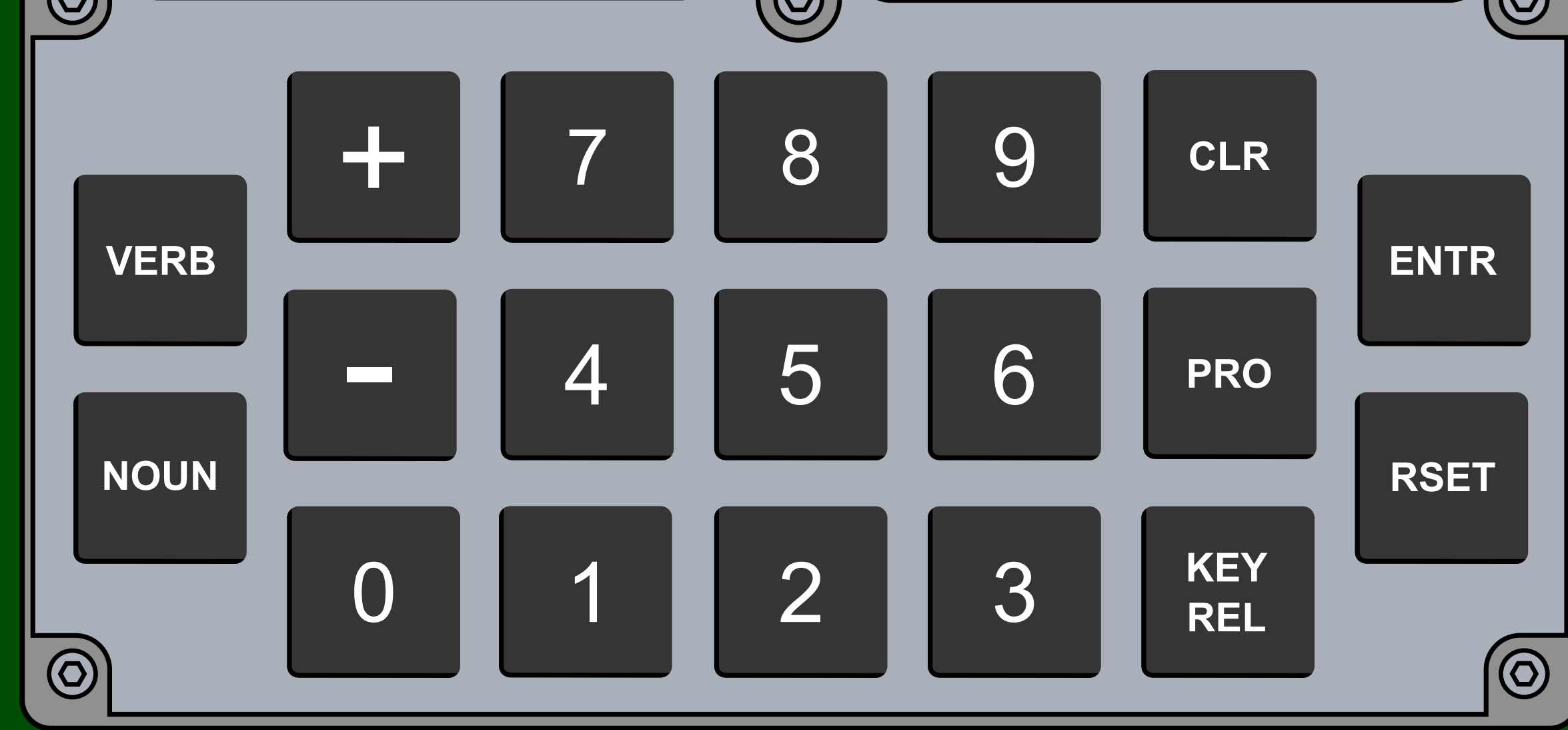
GOTO
CALL
RVQ
STQ
CGOTO
CCALL
BPL
BZE
BMN
BHIZ
EXIT
RTB
BOV
BOVB



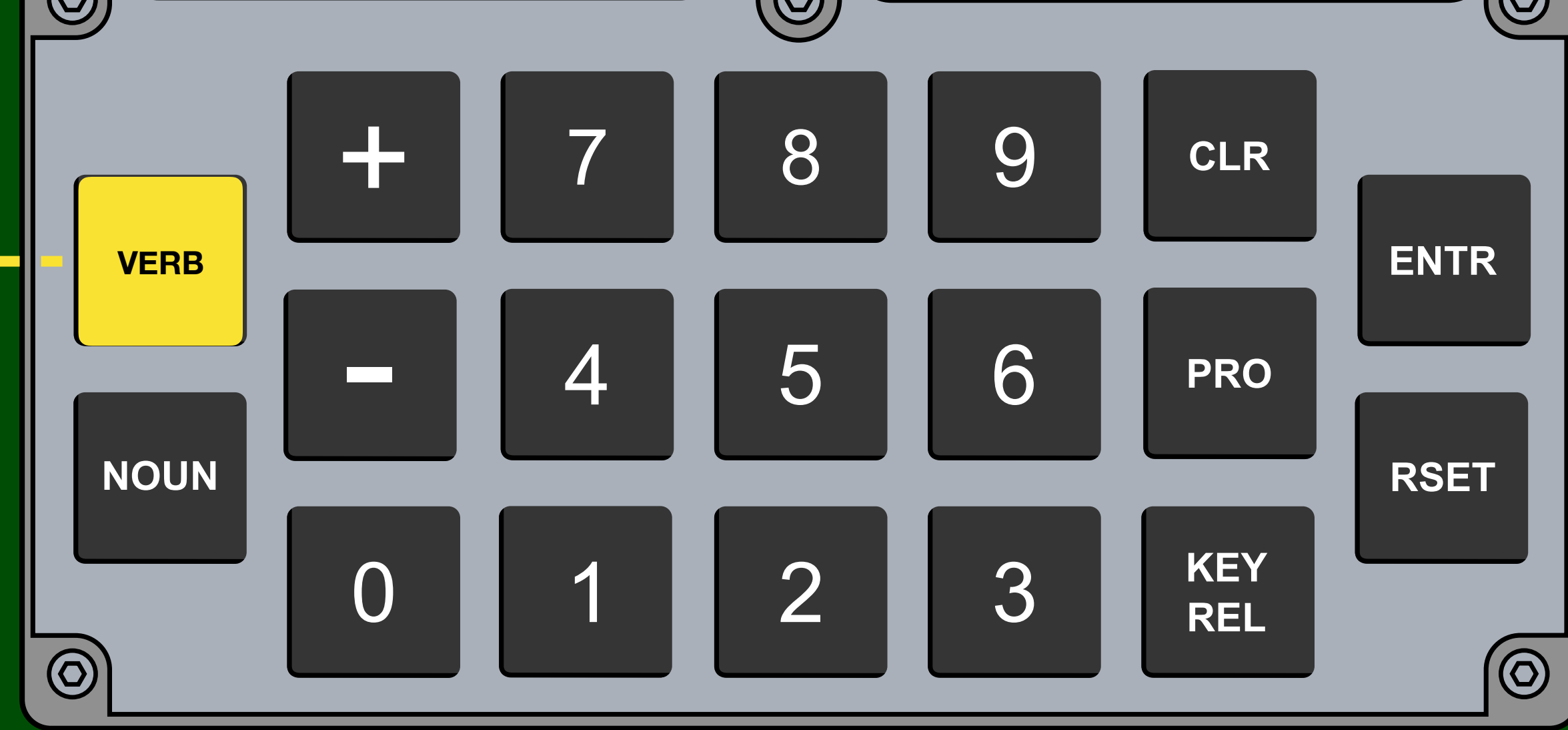
Interrupts



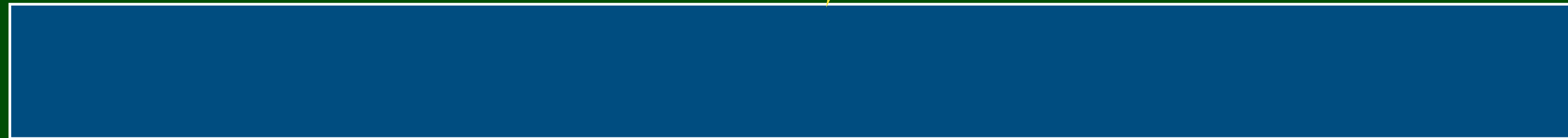
# Interrupts



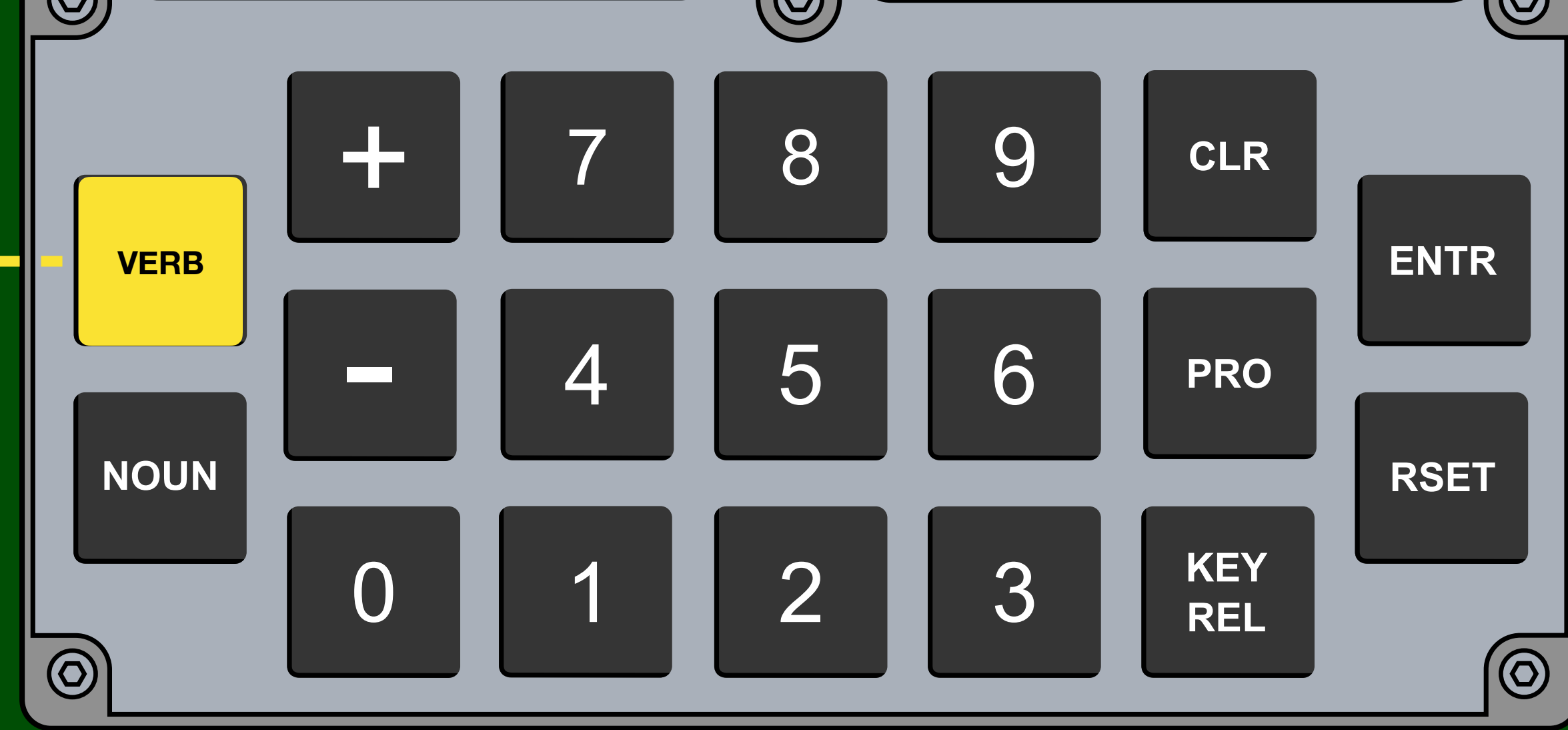
# Interrupts



Interrupt



# Interrupts

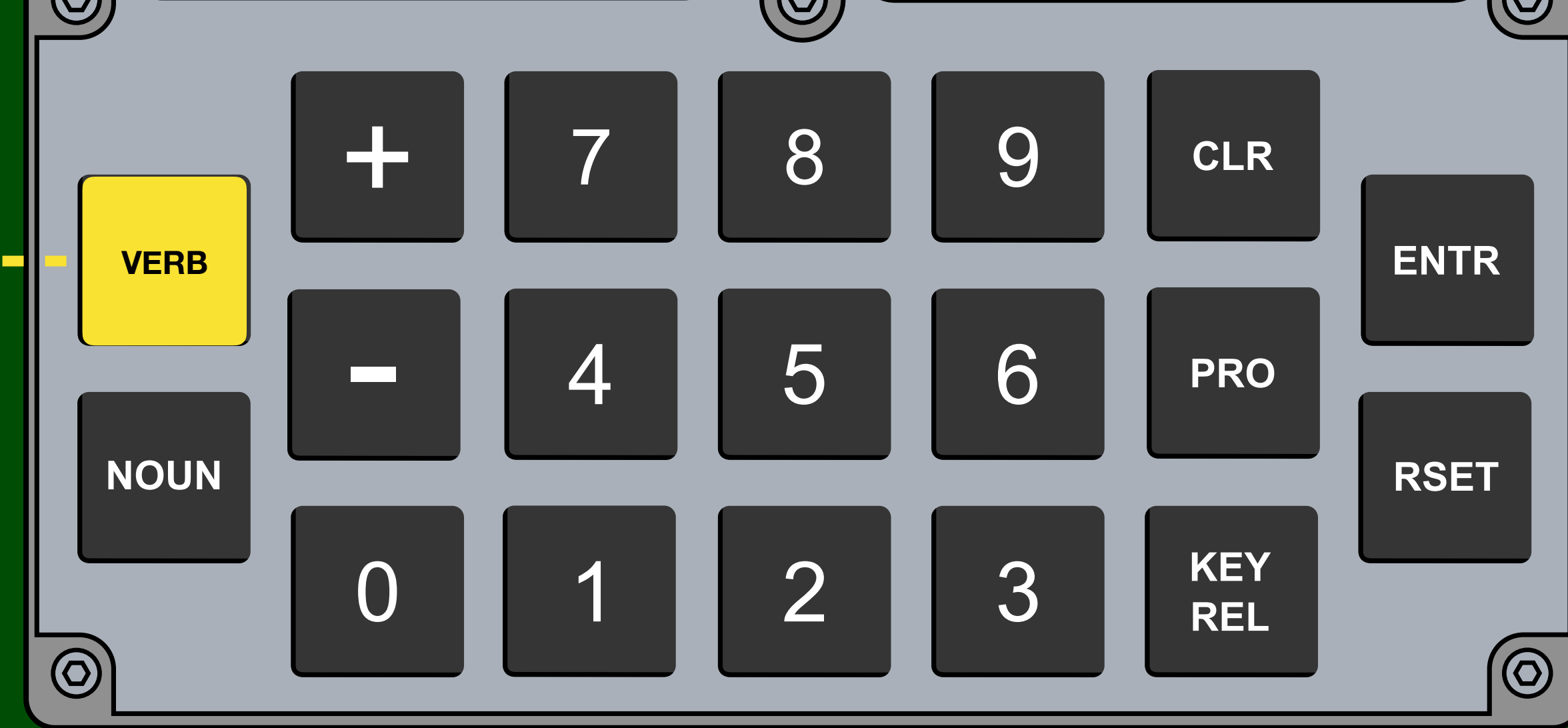


Interrupt





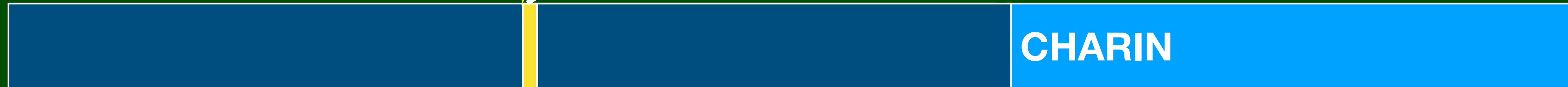
# Interrupts



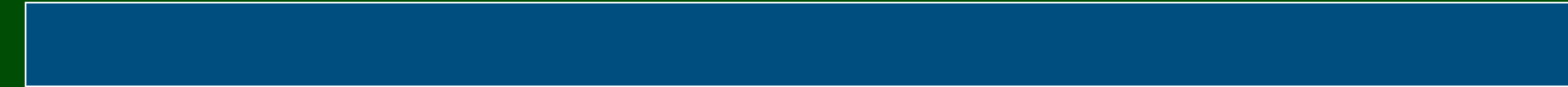
Interrupt



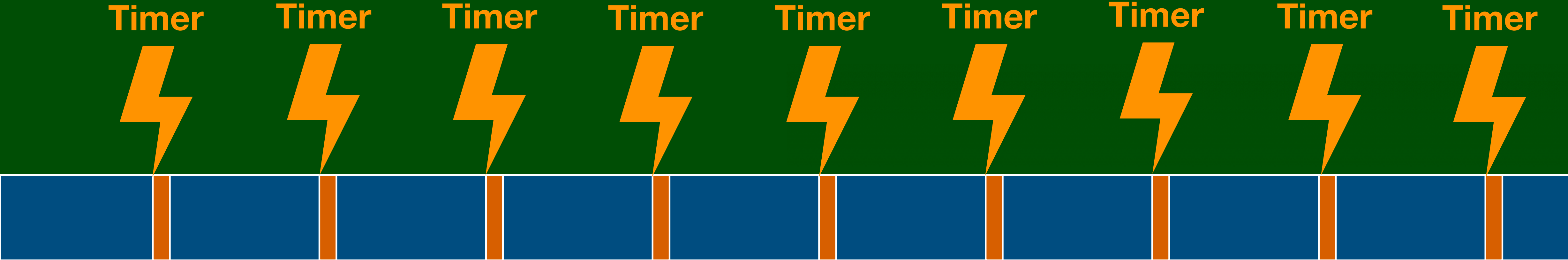
NOVAC  
create job



120 ms Timer



120 ms Timer



120 ms Timer





Drivers



Drivers

0

1

2

3

4

5

6

7

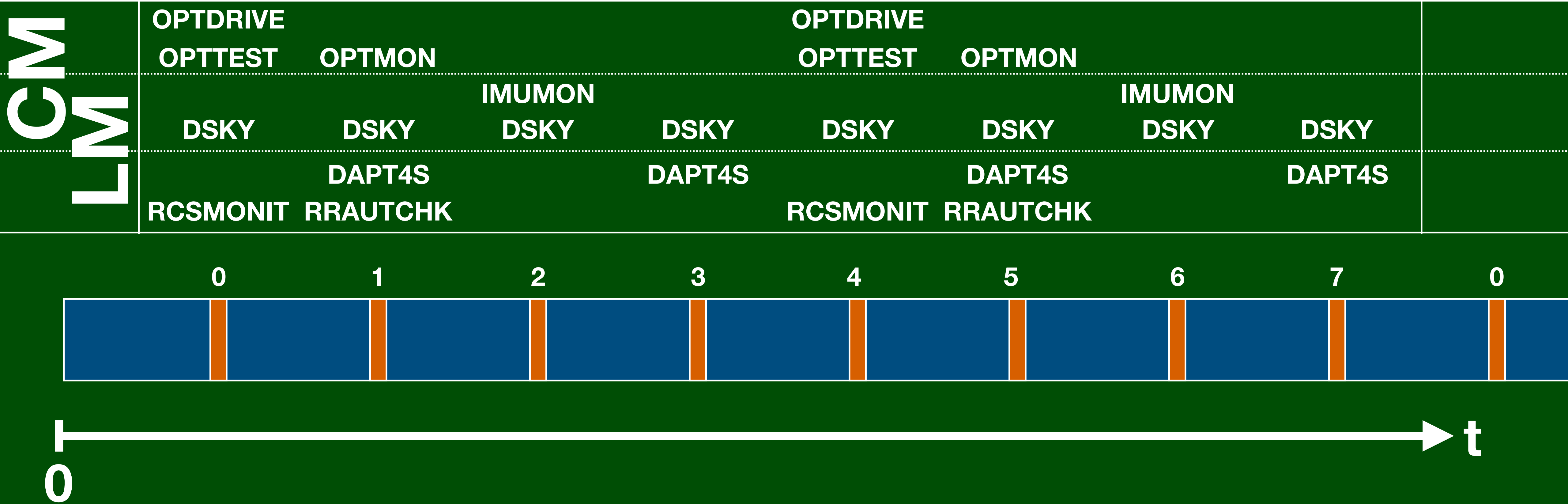
0

0

0

t

Drivers



# Waitlist Tasks



→ t



# Waitlist Tasks



Job A



# Waitlist Tasks



Job A

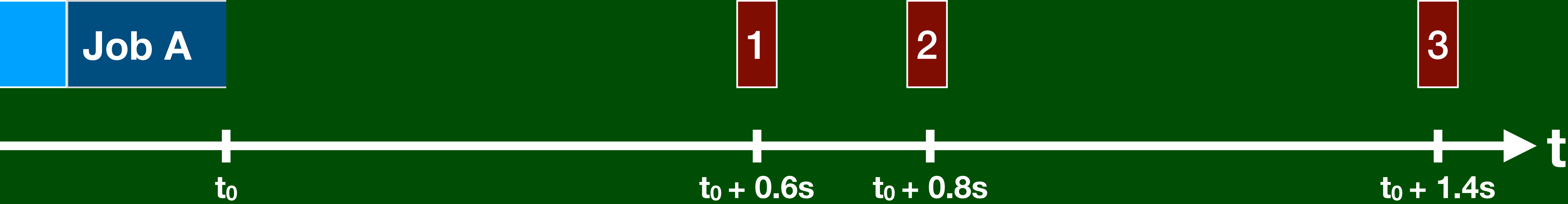


Waitlist Tasks

- Task 1 @ 0.6
- Task 2 @ 0.8
- Task 3 @ 1.4



Job A



Waitlist Tasks

- Task 1 @ 0.6
- Task 2 @ 0.8
- Task 3 @ 1.4





Waitlist Tasks

- Task 1 @ 0.6
- Task 2 @ 0.8
- Task 3 @ 1.4



Waitlist Tasks

- Task 1 @ 0.6
- Task 2 @ 0.8
- Task 3 @ 1.4



**Waitlist**

Waitlist

LST1	
0	
1	300
2	301
3	302
4	303
5	304
6	305
7	306
8	307



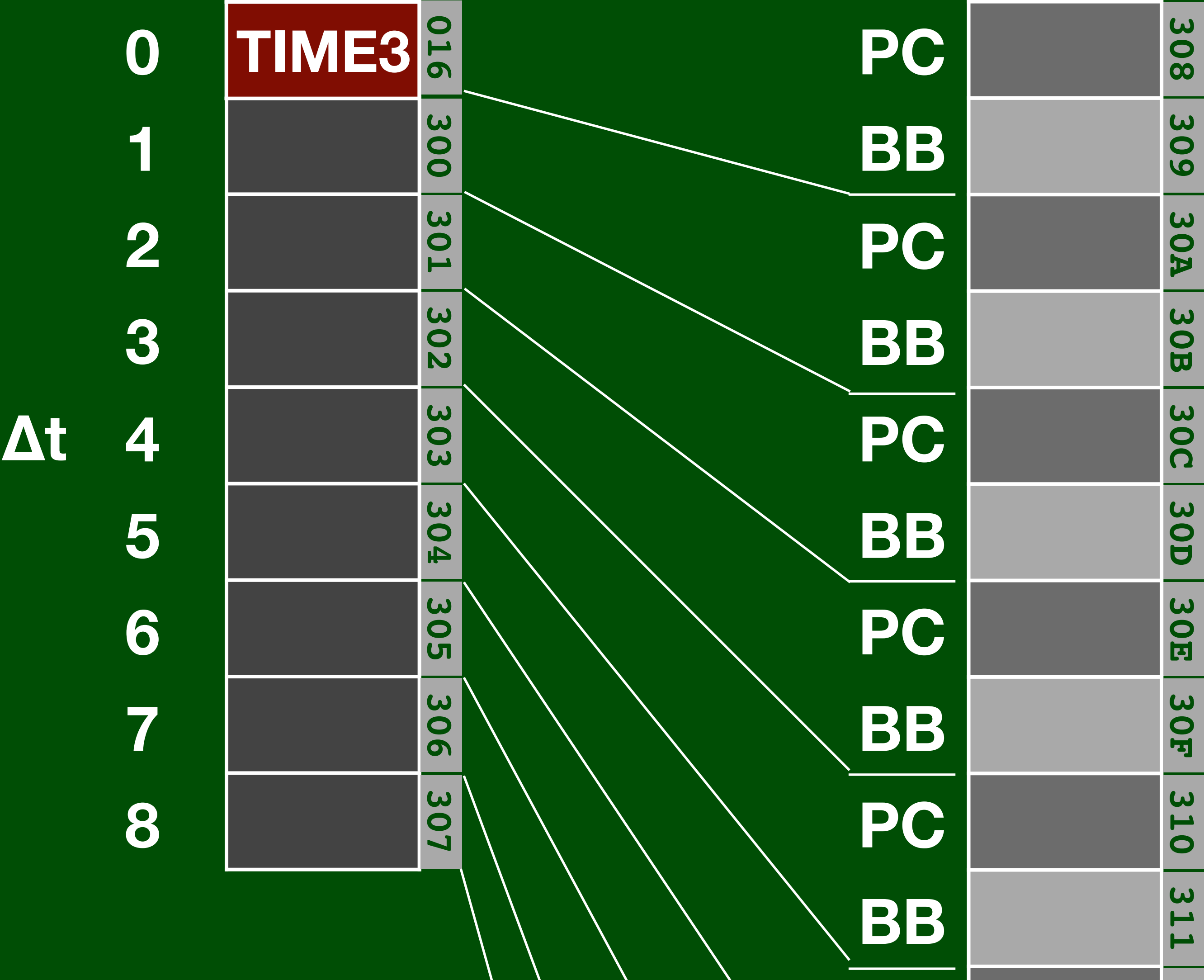
Waitlist

LST1		
$\Delta t$	0	TIME3016
	1	
	2	
	3	
	4	
	5	
	6	
	7	
	8	

Waitlist

LST1

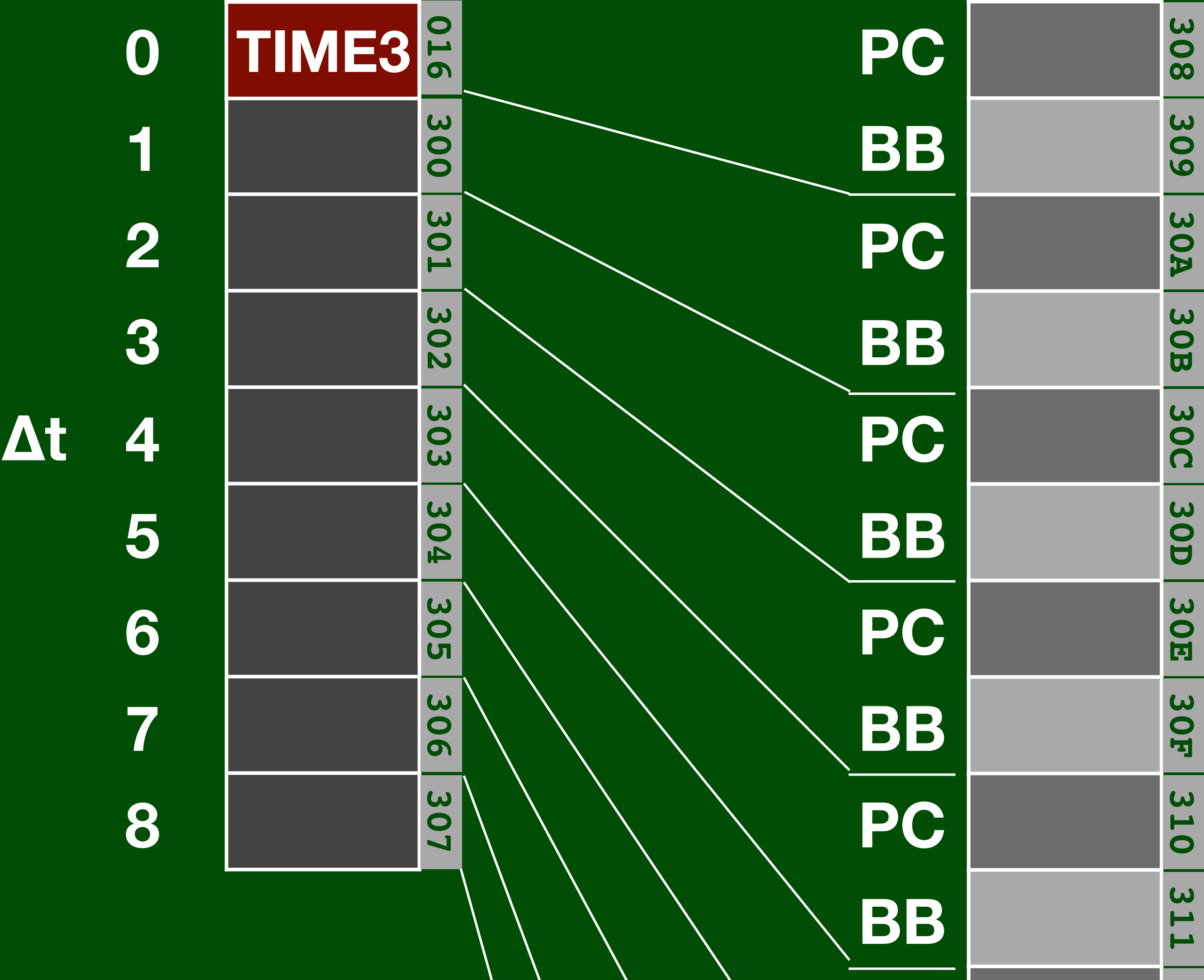
LST2



Waitlist

LST1

LST2



**WAITLIST**  
**TASKOVER**

Create new task  
End current task

Real-Time

t



Real-Time

every 10 ms

waitlist

TIME3

0000

016

+ 0 ms

3

3

10 ms

20 ms

t

Real-Time

every 10 ms

waitlist

TIME3

0000

016

+ 0 ms

autopilot

TIME5

0000

018

+ 5 ms

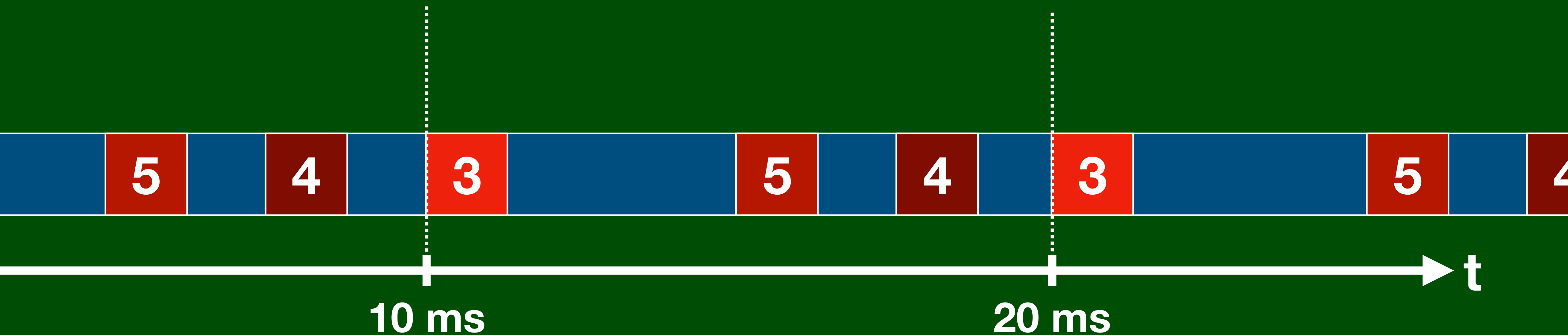
drivers

TIME4

0000

017

+ 7.5 ms



Real-Time

every 10 ms

waitlist **TIME3**

0000

016

+ 0 ms

autopilot **TIME5**

0000

018

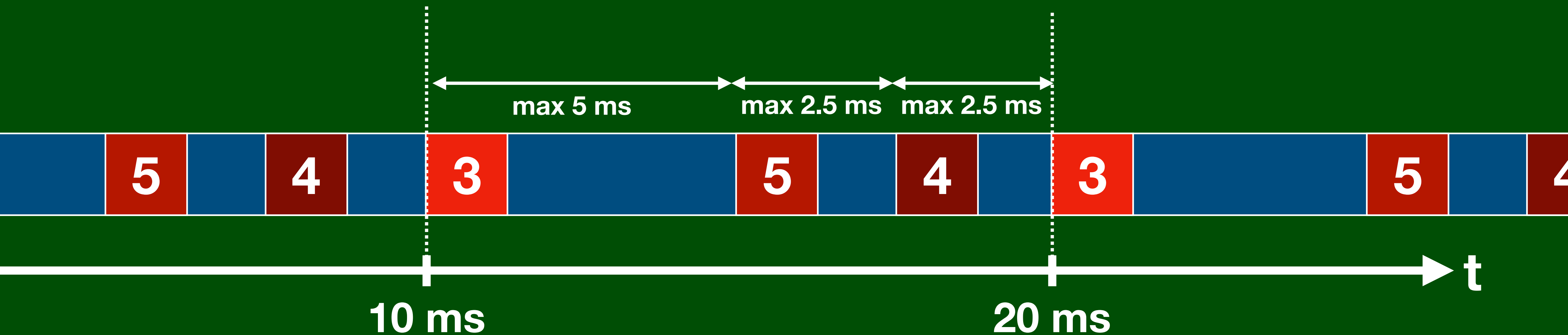
+ 5 ms

drivers **TIME4**

0000

017

+ 7.5 ms



Real-Time

every 10 ms

waitlist **TIME3**

0000

016

+ 0 ms

autopilot **TIME5**

0000

018

+ 5 ms

drivers **TIME4**

0000

017

+ 7.5 ms

Interrupt



max 5 ms

max 2.5 ms

max 2.5 ms

10 ms

20 ms

t





Real-Time

every 10 ms

waitlist **TIME3**

0000

016

+ 0 ms

autopilot **TIME5**

0000

018

+ 5 ms

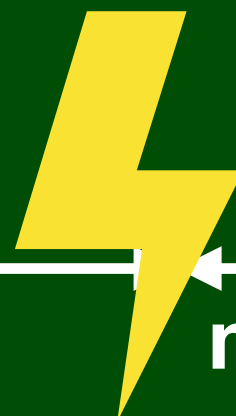
drivers **TIME4**

0000

017

+ 7.5 ms

Interrupt



max 5 ms

max 2.5 ms

max 2.5 ms

10 ms

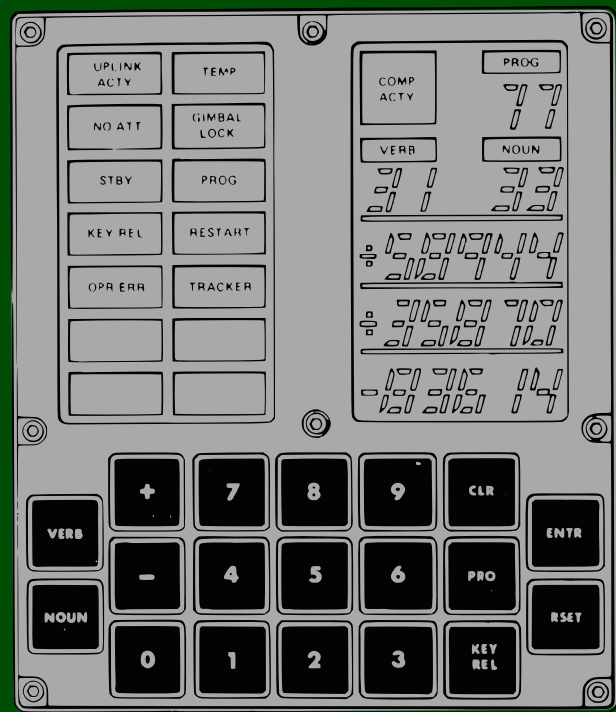
20 ms

t

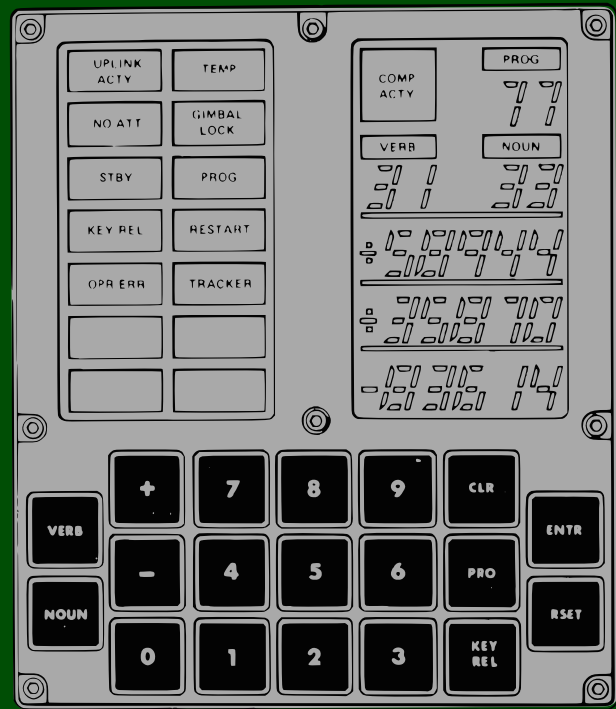


# PINBALL – the Shell

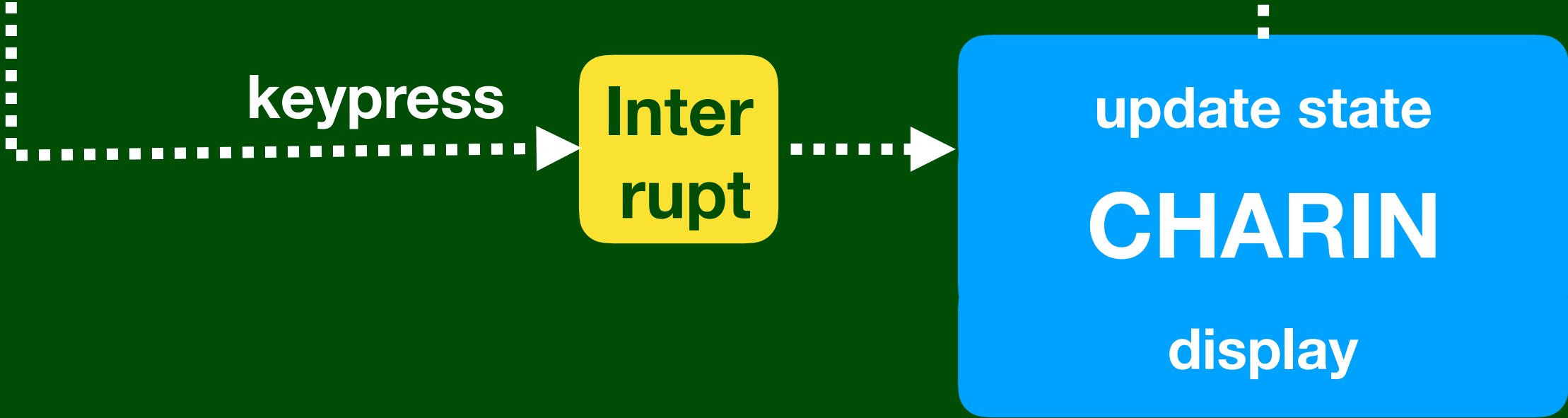
# PINBALL – the Shell



# PINBALL – the Shell



**VERBREG**  
**NOUNREG**

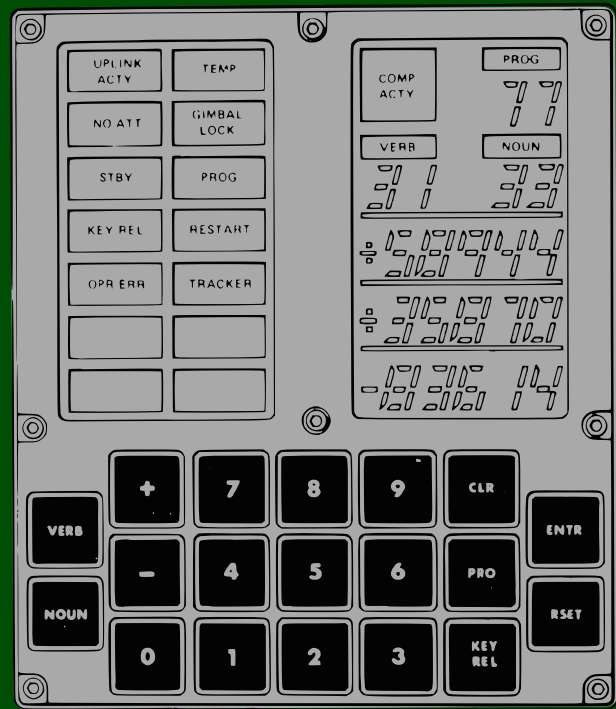


## DSPTAB

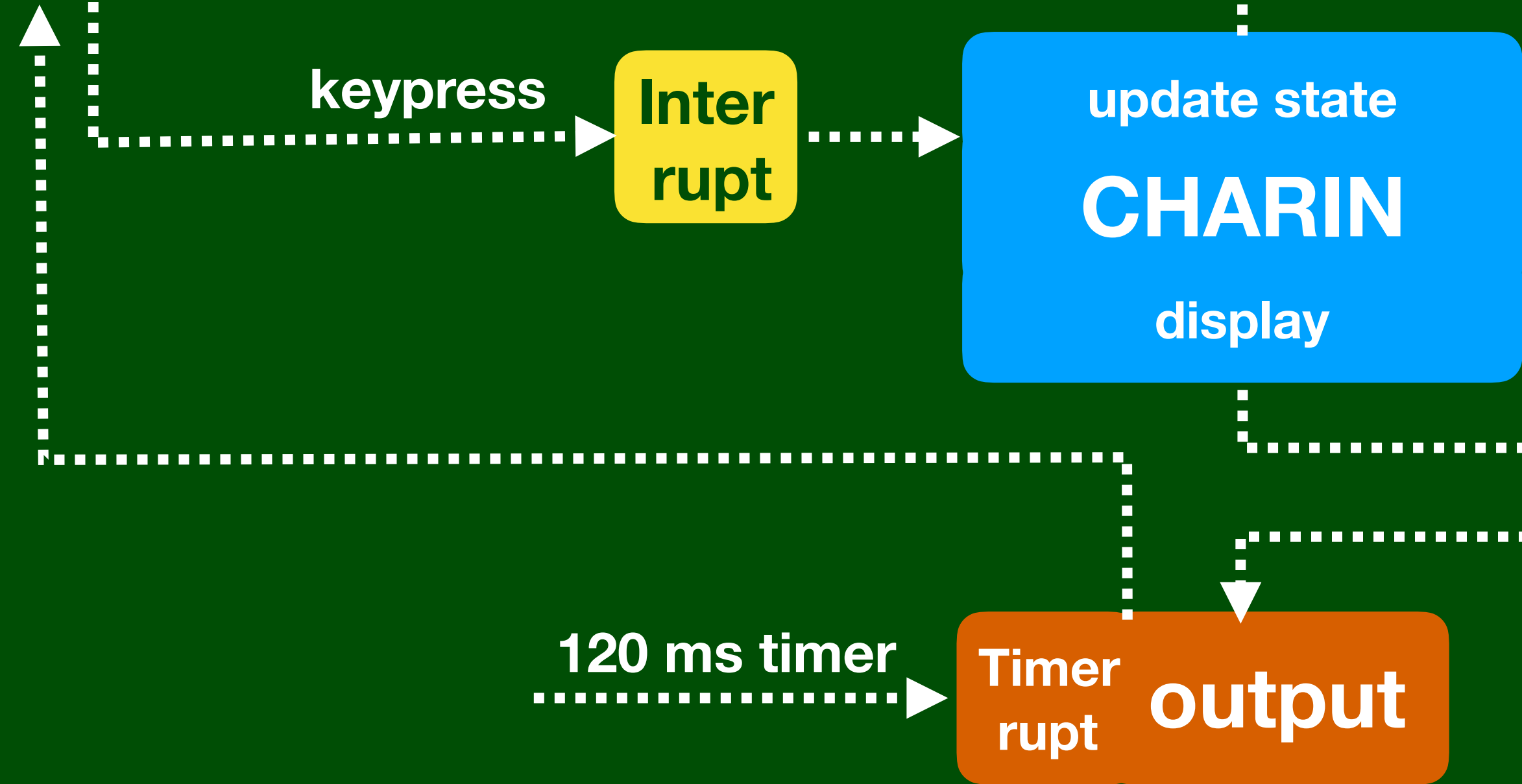
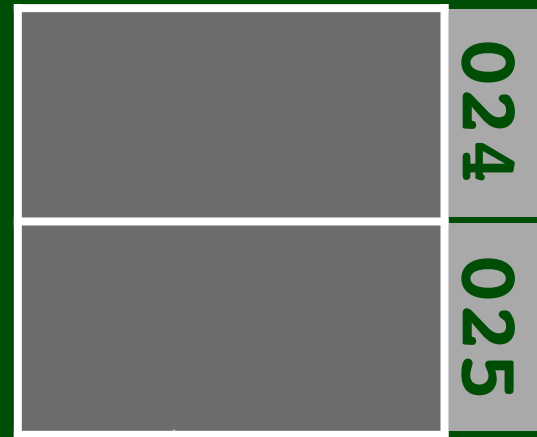
	212
	213
	214
	215
	215
	216
	217
	218
	219
	21A
	21B



# PINBALL – the Shell



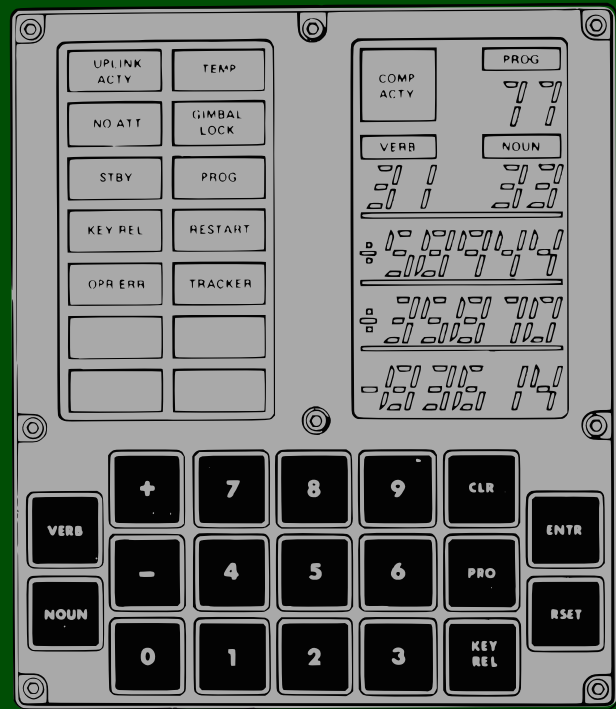
**VERBREG**  
**NOUNREG**



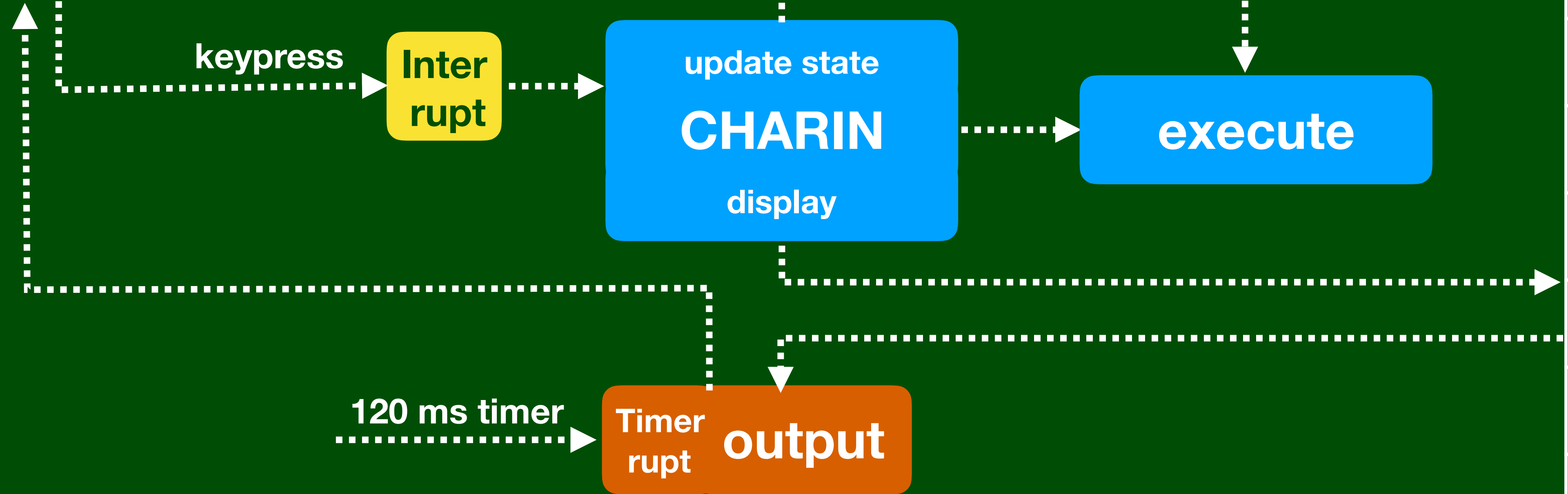
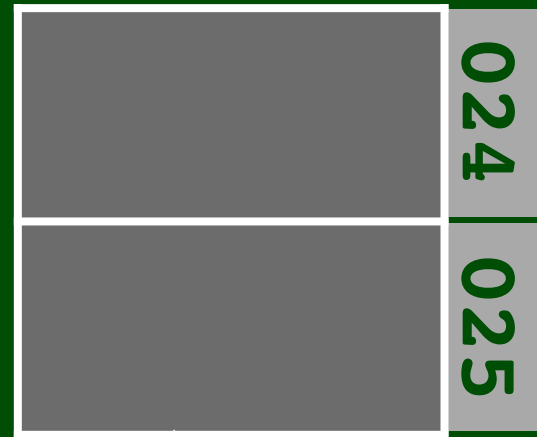
## DSPTAB

	212
	213
	214
	215
	215
	216
	217
	218
	219
	21A
	21B

# PINBALL – the Shell



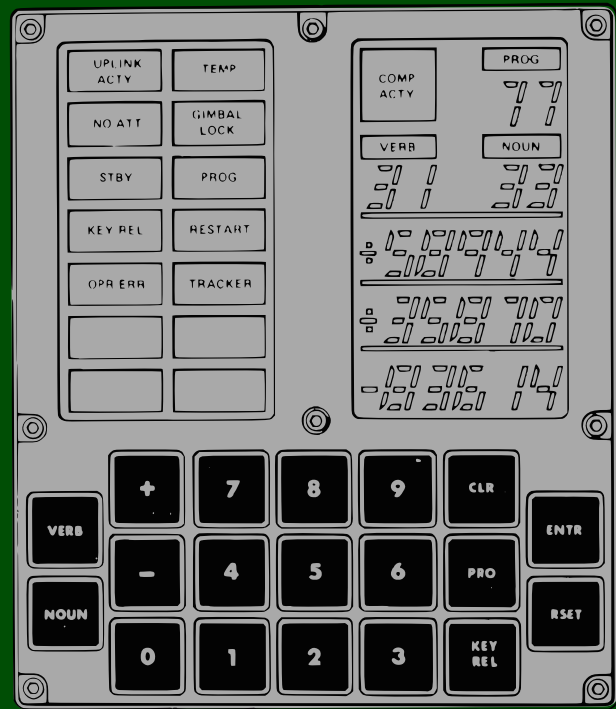
**VERBREG**  
**NOUNREG**



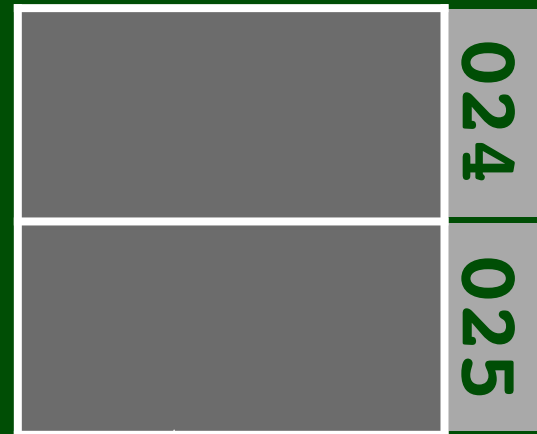
**DSPTAB**

	212
	213
	214
	215
	215
	216
	217
	218
	219
	21A
	21B

# PINBALL – the Shell



VERBREG  
NOUNREG



## DSPTAB

	212
	213
	214
	215
	215
	216
	217
	218
	219
	21A
	21B

keypress

Inter  
rupt

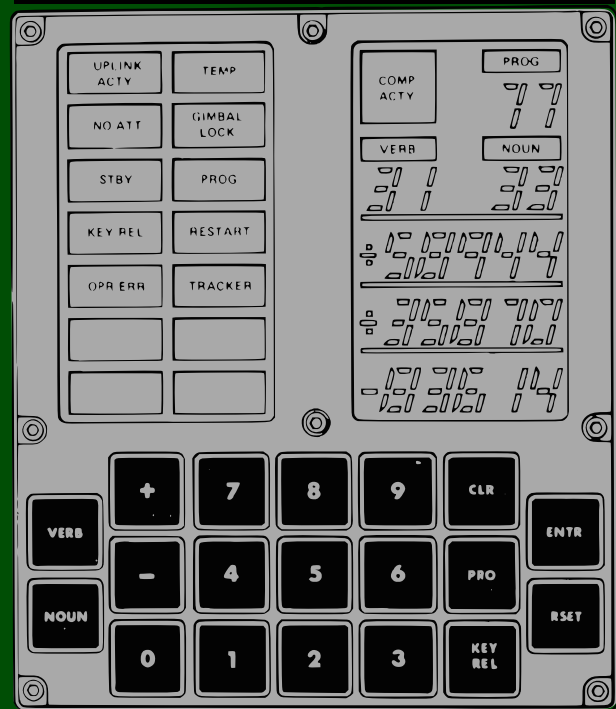
uplink

Inter  
rupt

update state  
CHARIN  
display

execute

Mission Control



Radio

120 ms timer

Timer  
rupt

output

20 ms timer

Down  
rupt

downlink

# Program Alarm

UPLINK  
ACTY

NO ATT

STBY

KEY REL

OPR ERR

TEMP

GIMBAL  
LOCK

PROG

RESTART

TRACKER

ALT

VEL

COMP  
ACTY

PROG

VERB

NOUN

VERB

NOUN

+

-

0

7

4

1

8

5

2

9

6

3

CLR

PRO

KEY  
REL

ENTR

RSET



Program Alarm

0000	E
0B77	41F
3282	420
	4

call ALARM  
.word 031202



Program Alarm

0000	E
0B77	41F
3282	420
	4

```
call ALARM
.word 031202
```



## Program Abort

Program Error

Program Error

Program Error

Program Error

Program Error

Program Error

Program Error

Program Error

Program Error

Program Error

Program Error

Program Error

Program Error

Program Error

Program Error

Program Error

Program Error

Program Abort

Interrupt Watchdog

JMP Watchdog

NEWJOB Watchdog

Parity

0000	E
0B77	41F
0242	420
0000	4

`call ABORT`

`.word 01102`



Program Abort

Interrupt Watchdog

JMP Watchdog

NEWJOB Watchdog

Parity

RESET



0000	E
0B77	41F
0242	420
0000	4

```
call ABORT
.word 01102
```

Program Abort

Job 4

Interrupt Watchdog

JMP Watchdog

NEWJOB Watchdog

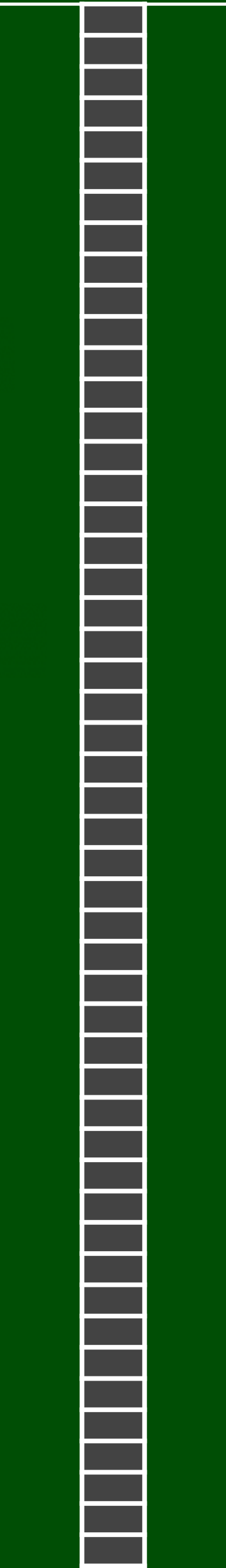
Parity

RESET



0000	E
0B77	41F
0242	420
0000	4

```
call ABORT
.word 01102
```



Program Abort

Interrupt Watchdog

JMP Watchdog

NEWJOB Watchdog

Parity

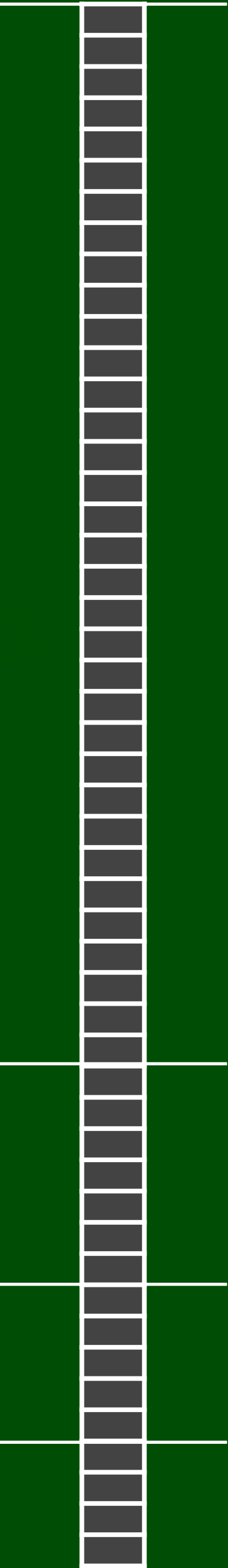
RESET



0000	E
0B77	41F
0242	420
0000	4

```
call ABORT
.word 01102
```

Job 4



Program Abort

Interrupt Watchdog

JMP Watchdog

NEWJOB Watchdog

Parity

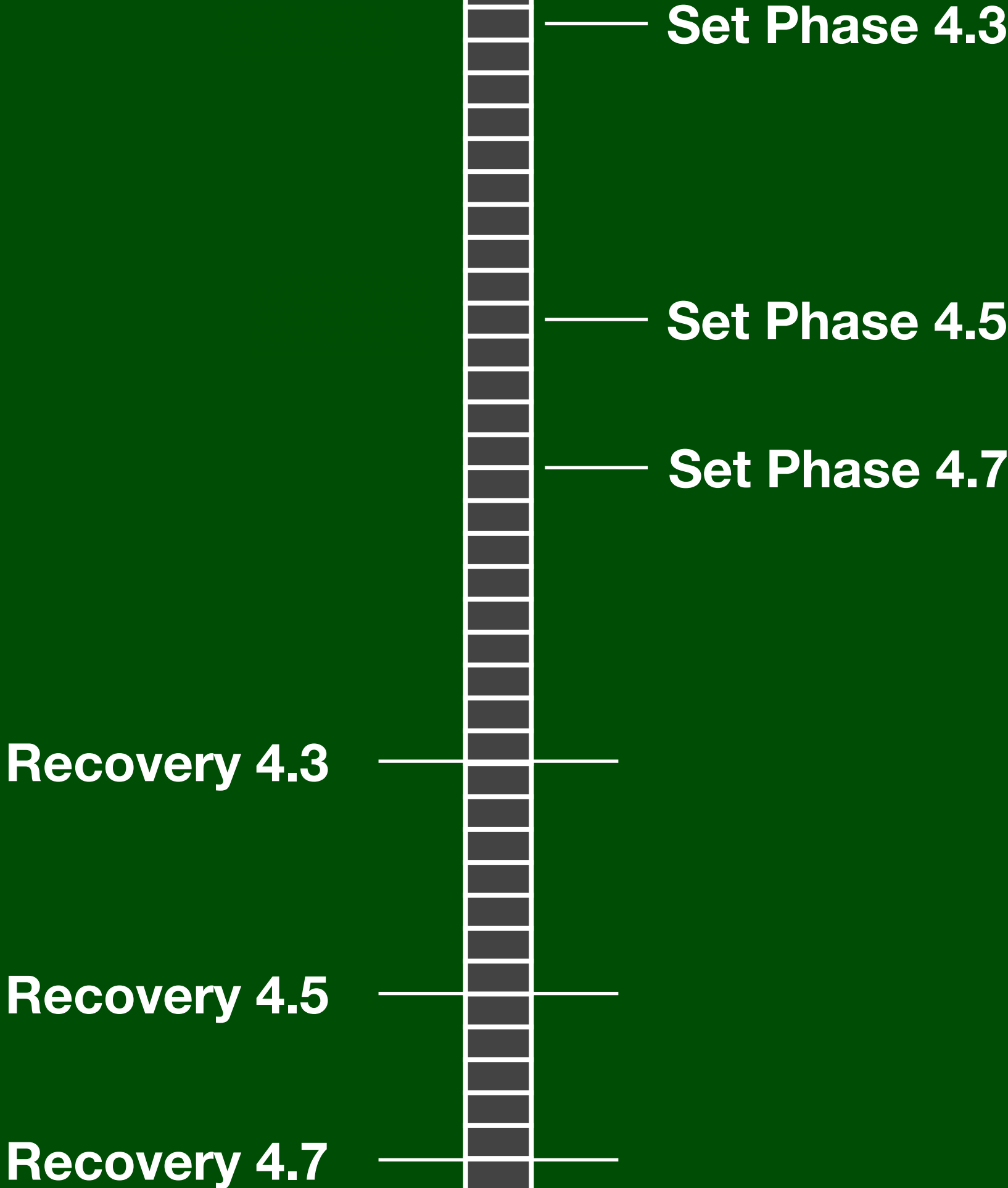
RESET



0000	E
0B77	41F
0242	420
0000	4

```
call ABORT
.word 01102
```

Job 4





Program Abort

Interrupt Watchdog

JMP Watchdog

NEWJOB Watchdog

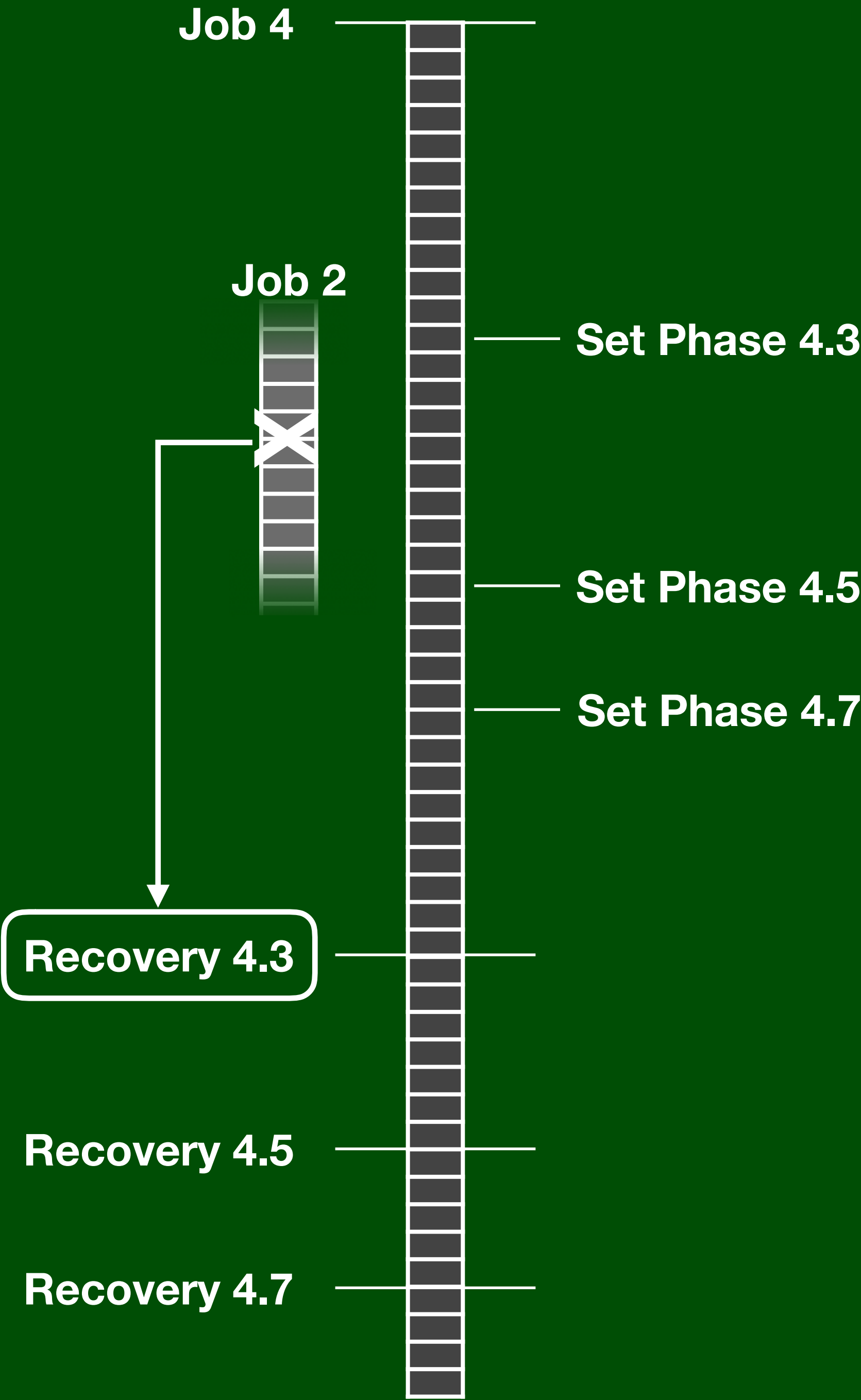
Parity

RESET



0000	E
0B77	41F
0242	420
	4

```
call ABORT
.word 01102
```



Program Abort

Interrupt Watchdog

JMP Watchdog

NEWJOB Watchdog

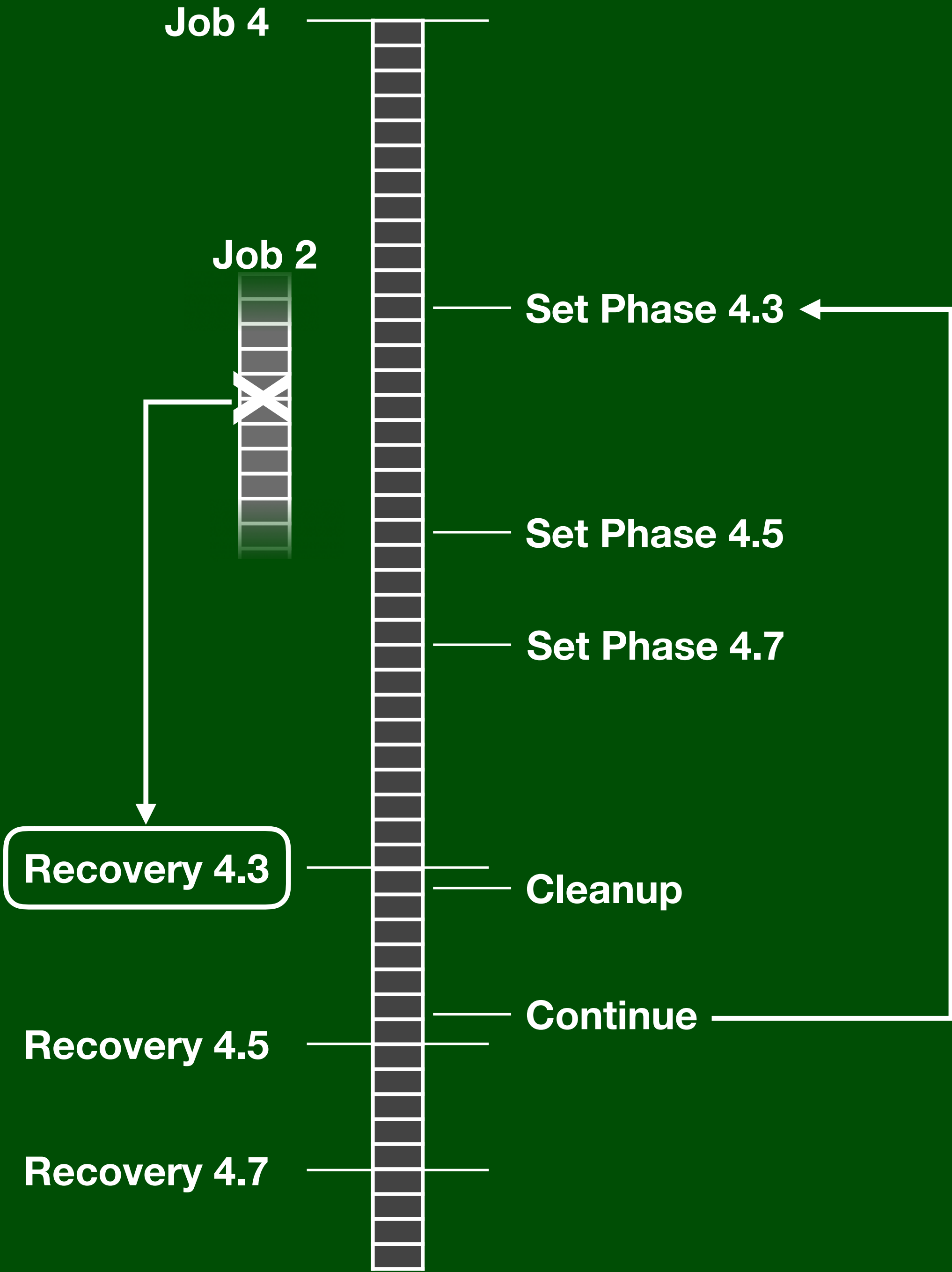
Parity

RESET



0000	E
0B77	41F
0242	420
	4

```
call ABORT
.word 01102
```



Program Abort

Interrupt Watchdog

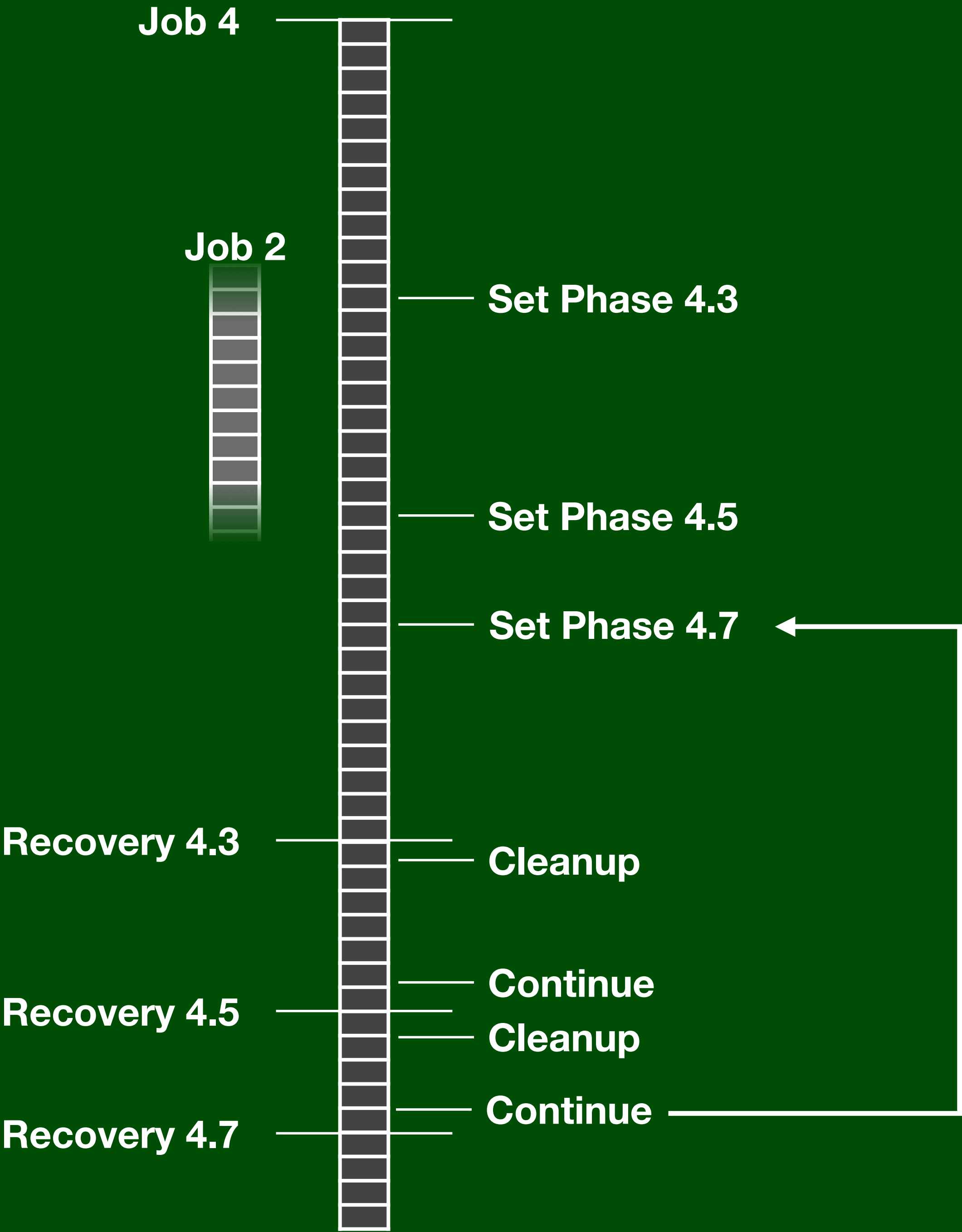
JMP Watchdog

NEWJOB Watchdog

Parity

0000	E
0B77	41F
0242	420
0000	4

```
call ABORT
.word 01102
```



Program Abort

Interrupt Watchdog

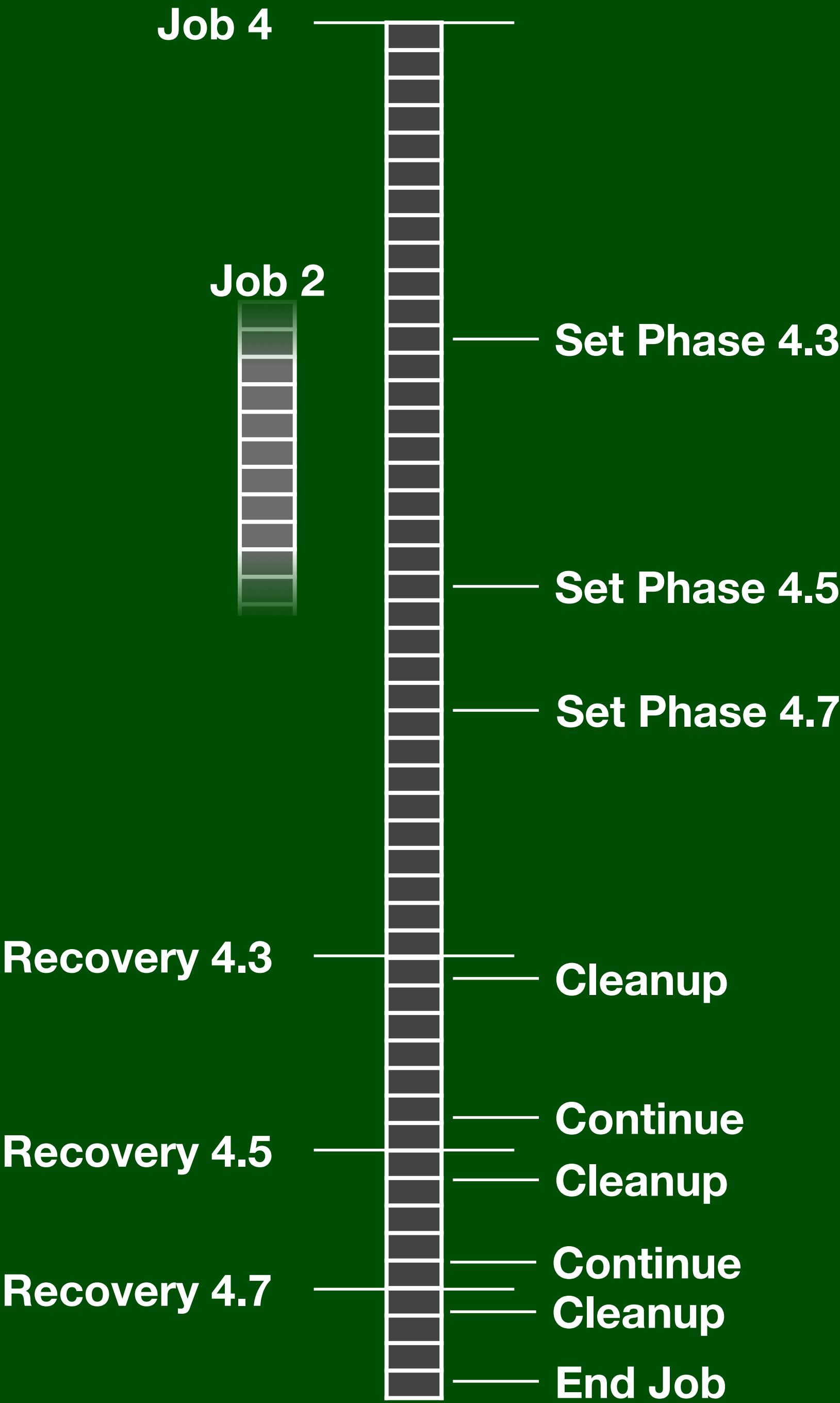
JMP Watchdog

NEWJOB Watchdog

Parity

0000	E
0B77	41F
0242	420
	4

```
call ABORT
.word 01102
```





# Program Abort

Program Abort

Program Abort

Program Abort

Program Abort

0000	E
0B77	41F
002C	420
0000	4

call PHASCHNG

.word 00054

Recovery  
Table

Job	
1	
2	
3	
4	
5	
6	

Program Abort

0000	E
0B77	41F
002C	420
0000	4

call PHASCHNG

.word 00054



# Recovery Table

Job	
1	
2	
3	
4	0005
5	
6	

Program Abort

0000	E
0B77	41F
002C	420
0000	4

call PHASCHNG

.word 00054



Recovery Table

Job	
1	
2	
3	
4	0005
5	
6	

4.3

Time	Task
	PC
	BB



Program Abort

00000	E
0B77	41F
002C	420
00000	4

call PHASCHNG

.word 00054



Recovery Table

Job	
1	
2	
3	
4	0005
5	
6	

		Time	Task
4.3		PC	
		BB	
		Priority	Job
4.5		PC	
		BB	

Program Abort

00000	E
0B77	41F
002C	420
00000	4

call PHASCHNG

.word 00054



Recovery Table	
Job	
1	
2	
3	
4	0005
5	
6	

		Time	Task	
4.3	-	PC		
		BB		
	-	Priority	Job	
	+	PC		
4.5		BB		
4.7	+	Priority	VAC Job	
	+	PC		
		BB		

(A second table with even phases contains 2 jobs/tasks each. Phase 0 disables recovery and Phase 1 disables recovery and retains the display contents.)

## Program Abort

41E

420

**.word 00054**

## Job

[illegible]

## Job

6

0005

## 4.7

BB

**Job VAC**

**(A second table with even phases contains 2 jobs/tasks each. Phase 0 disables recovery and Phase 1 disables recovery and retains the display contents.)**

## Program Abort

# 41E

4204

**.word 00054**

## Job

The diagram consists of 10 vertical bars arranged horizontally. Each bar is composed of 10 equal-sized rectangular segments stacked vertically. The first bar on the left is entirely red. The remaining 9 bars each have 9 dark gray segments and 1 red segment at the bottom. The bars are set against a green background with a white border.



## Job 4

## Job

6

0005

## 4.7

BB

**Job VAC**

**(A second table with even phases contains 2 jobs/tasks each. Phase 0 disables recovery and Phase 1 disables recovery and retains the display contents.)**



## Program Abort

# 41E

420 4

**.word 00054**

# Job

The diagram consists of 10 vertical bars arranged horizontally. Each bar is composed of 10 equal segments. The first bar on the left is entirely red. The remaining 9 bars each have 9 dark gray segments and 1 red segment at the bottom. The bars are set against a green background with a white grid.

## Job 4

## Job

0005	

3

4

5

6

4.5	-	Priority
	+	PC
		BB
		Job

+

+

Priority

PC

BB

VAC

Job

**(A second table with even phases contains 2 jobs/tasks each. Phase 0 disables recovery and Phase 1 disables recovery and retains the display contents.)**

## Program Abort

# 41E

420

**.word 00054**

# Job

The diagram consists of a 10x10 grid of squares. The first column contains 9 blue squares and 1 green square at the bottom. The next seven columns each contain 9 dark gray squares and 1 red square at the bottom. The final column is empty. The entire grid is set against a green background.



## Job

6



## 4.7

BB

**Job VAC**

# Recovery Job 4.5

**(A second table with even phases contains 2 jobs/tasks each. Phase 0 disables recovery and Phase 1 disables recovery and retains the display contents.)**

**System Software**

**Multitasking**

**Shell**

**Interpreter**

**Fault Recovery**

**Device Drivers**

**Waitlist**

**System Software**



# Apollo Guidance Computer

Architecture

Mission Software

Hardware

Peripherals

System Software

VCF West

2019-08-04



Michael Steil

# The Apollo Guidance Computer



@pagetable

<http://www.pagetable.com/>





**Michael Steil**  
@pagetable



**Christian Hessmann**  
@hessi



**Dominik Wagner**  
@monkeydom



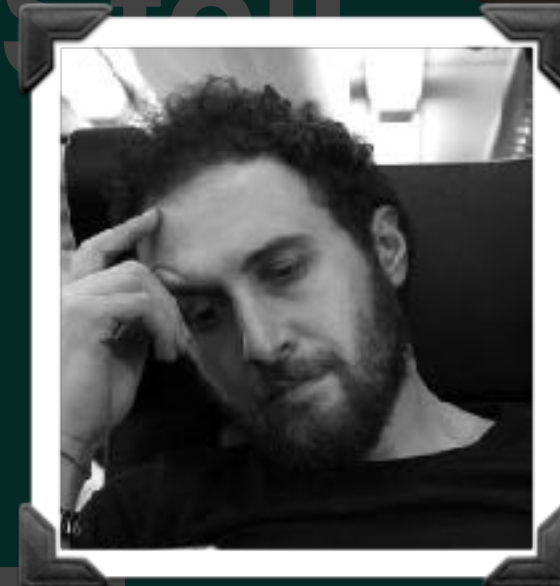
**Anna-Lena Baecker**  
@AnnLe\_\_



**Lisa Brodner**  
@ellduin



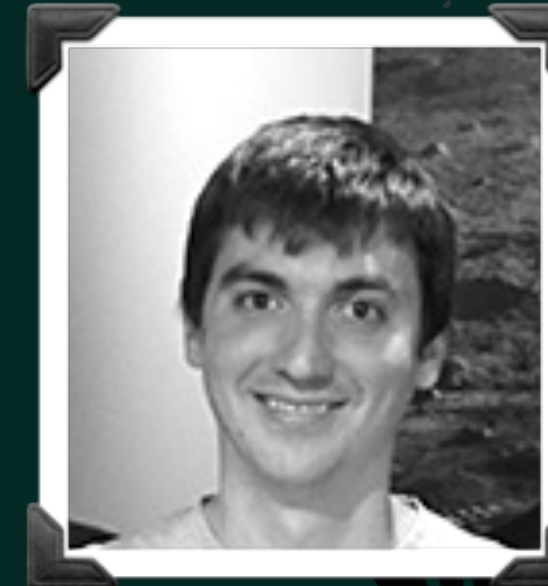
**Ange Albertini**  
<http://github.com/corkami/pics>



**Gianluca Guida**  
<http://www.tlbflush.org>



**Sven Oliver Moll**  
<https://xayax.net>



**Mike Stewart**  
[mikestewart.hcoop.net](http://mikestewart.hcoop.net)



**Frank O'Brien**  
<https://www.hq.nasa.gov/alsj/franko.html>



VCF West

2019-08-04



Michael Steil

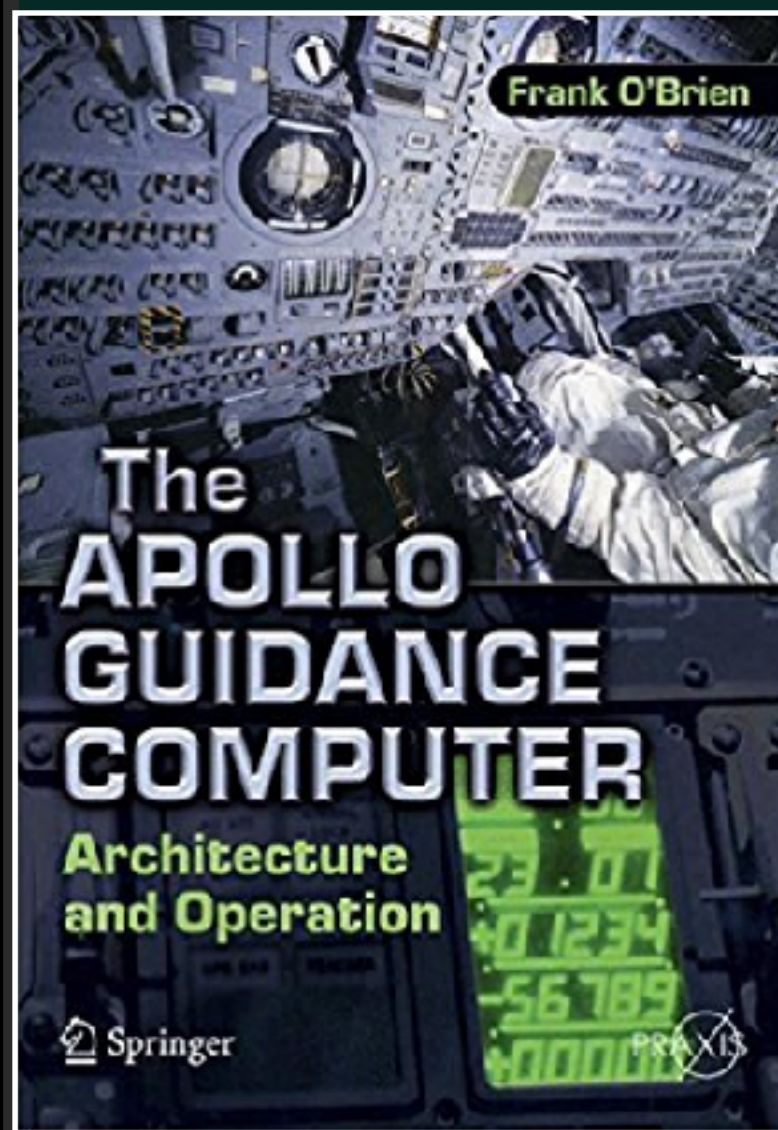
# The Apollo Guidance Computer



@pagetable

<http://www.pagetable.com/>





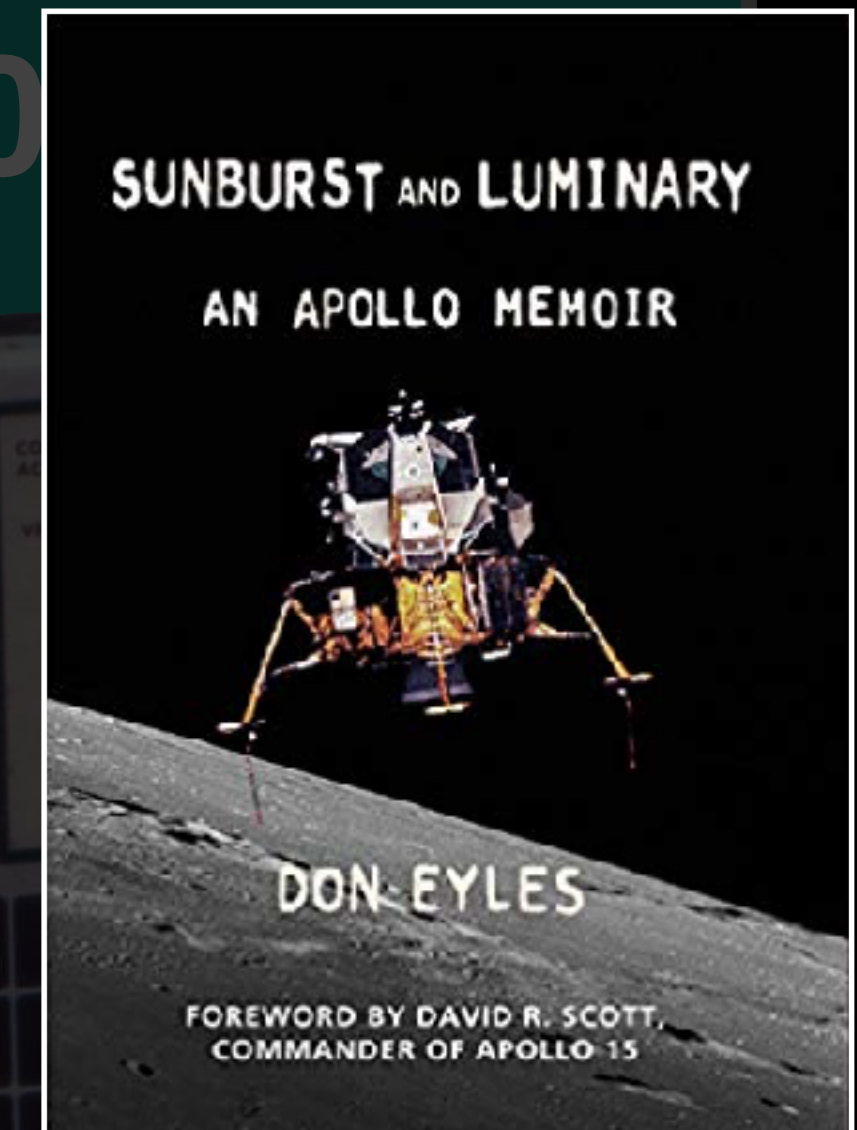
Frank O'Brien  
The Apollo Guidance Computer



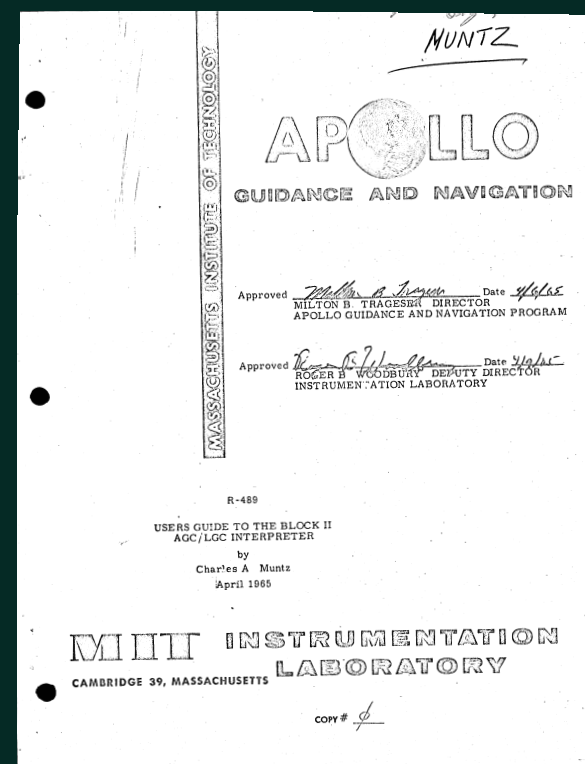
David A. Mindell  
Digital Apollo



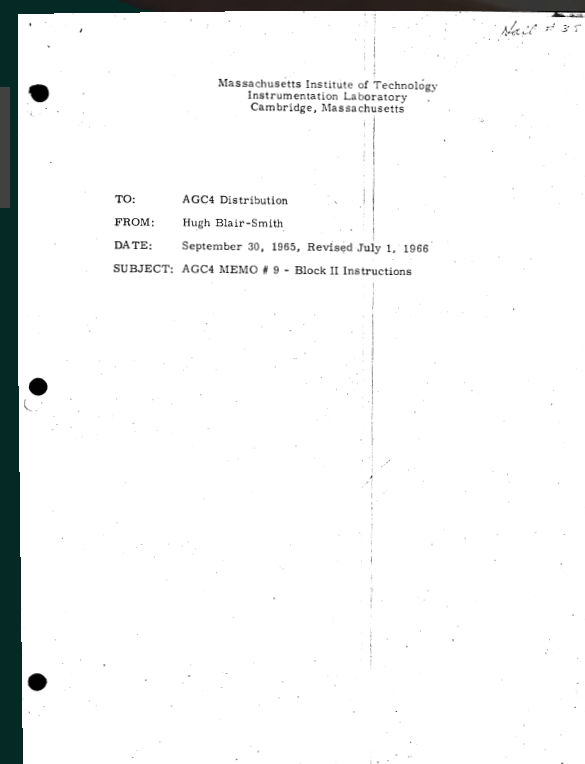
Eldon C. Hall  
Journey to the Moon



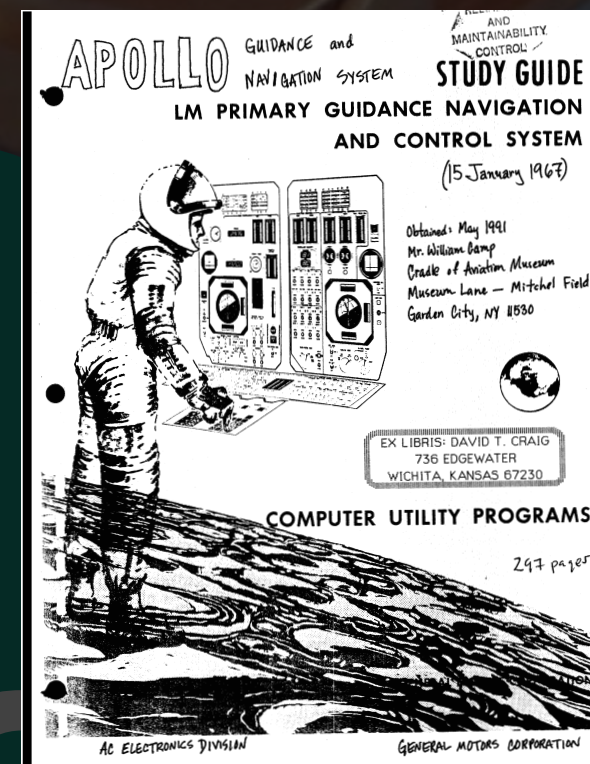
Don Eyles  
Sunburst and Luminary



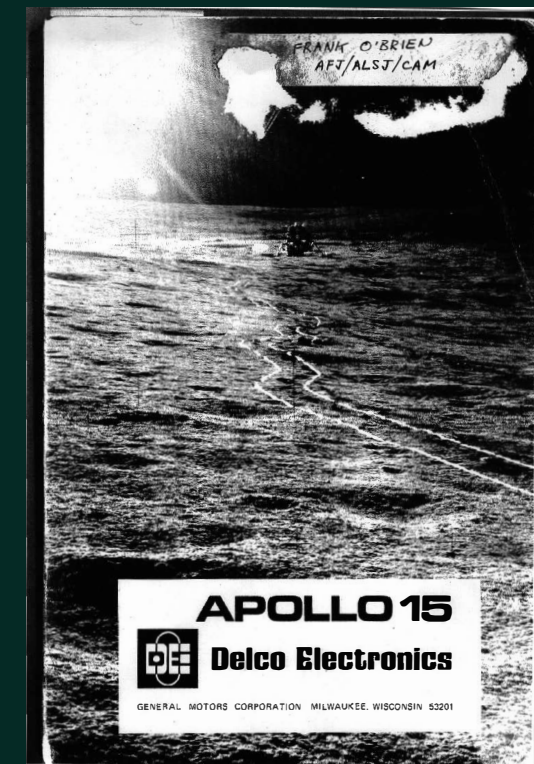
1687.pdf



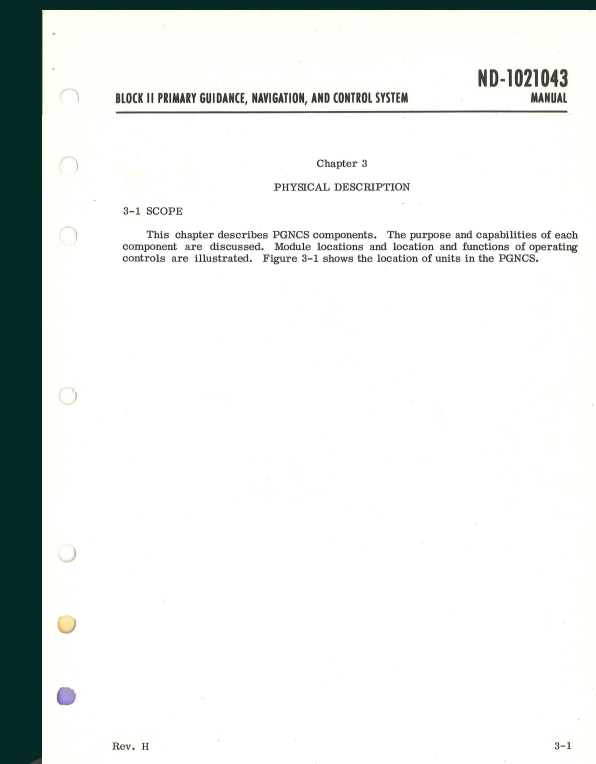
1689.pdf



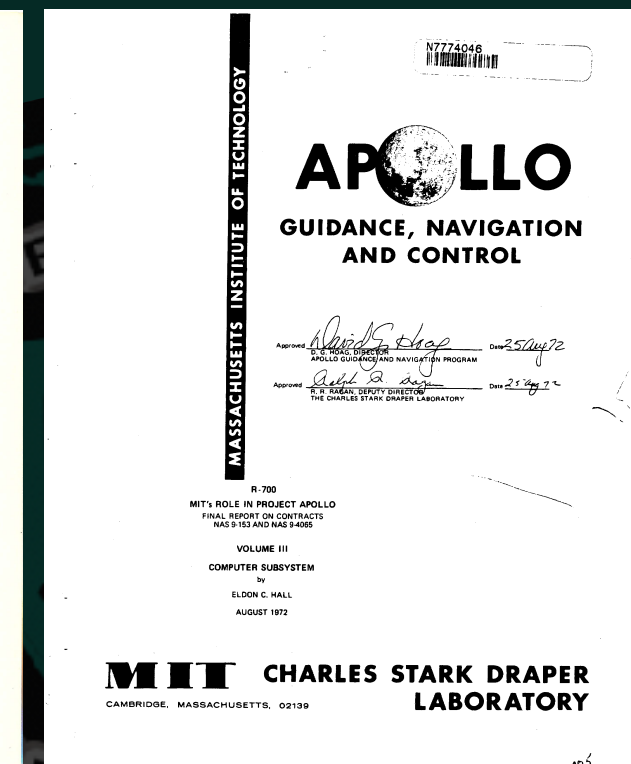
1709.pdf



A15Delco.pdf



HSI-208435-002.pdf



mitsrolevoliii.pdf

[Home \(AGC\)](#)  
[AGS](#)  
[LVDC](#)  
[Gemini](#)  
[Downloads](#)  
[Document library](#)  
[Change log](#)  
[Bug and issues](#)  
[Developer info](#)  
[Code repository](#)  
[Archival scans](#)

Virtual AGC — AGS — LVDC — Gemini

HOME PAGE  
Project Overview

<https://www.ibiblio.org/apollo/>

[FAQ](#)  
[yaAGC](#)  
[yaYUL](#)  
[yaDSKY](#)  
[yaOtherStuff](#)  
[Luminary](#)  
[Colossus](#)  
[Block 1 AGC](#)  
[Block 2 AGC Language](#)  
[Physical Implementations](#)  
[Do It Yourself](#)

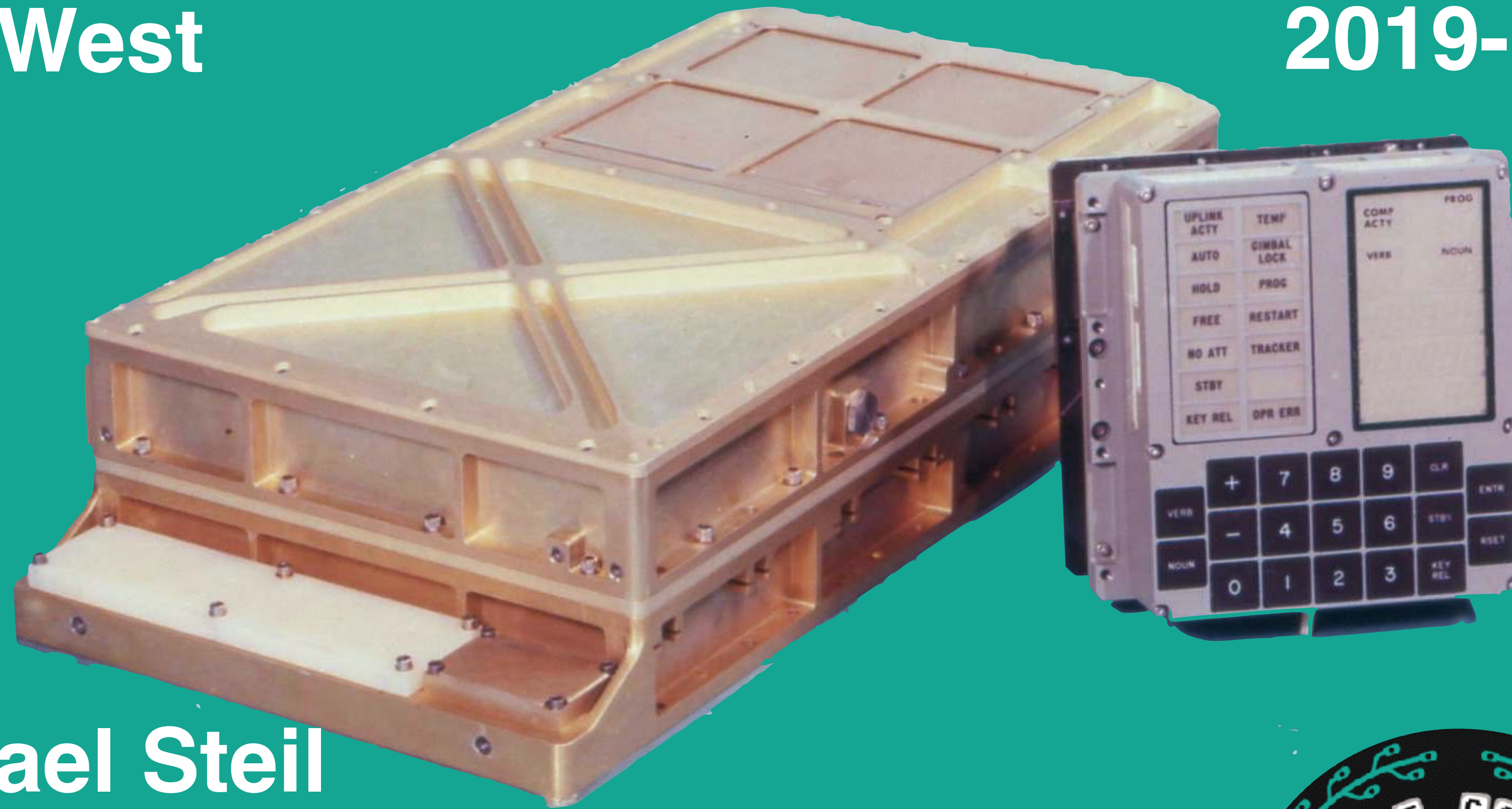


@pagetable  
[pagetable.com/](http://pagetable.com/)



VCF West

2019-08-04



Michael Steil

# The Apollo Guidance Computer



@pagetable

<http://www.pagetable.com/>