Computer Systems

Fifth edition
Documentation for Exam Handouts

J. Stanley Warford

March 22, 2016

Instruction Specifier	Mnemonic	Instruction	Addressing Mode	Status Bits
0000 0000	`STOP`	Stop execution	U	
0000 0001	`RET`	Return from `CALL`	U	
0000 0010	`RETTR`	Return from trap	U	
0000 0011	`MOVSPA`	Move SP to A	U	
0000 0100	`MOVFLGA`	Move NZVC flags to A⟨12..15⟩	U	
0000 0101	`MOVAFLG`	Move A⟨12..15⟩ to NZVC flags	U	
0000 011r	`NOTr`	Bitwise invert r	U	NZ
0000 100r	`NEGr`	Negate r	U	NZV
0000 101r	`ASLr`	Arithmetic shift left r	U	NZVC
0000 110r	`ASRr`	Arithmetic shift right r	U	NZC
0000 111r	`ROLr`	Rotate left r	U	C
0001 000r	`RORr`	Rotate right r	U	C
0001 001a	`BR`	Branch unconditional	i, x	
0001 010a	`BRLE`	Branch if less than or equal to	i, x	
0001 011a	`BRLT`	Branch if less than	i, x	
0001 100a	`BREQ`	Branch if equal to	i, x	
0001 101a	`BRNE`	Branch if not equal to	i, x	
0001 110a	`BRGE`	Branch if greater than or equal to	i, x	
0001 111a	`BRGT`	Branch if greater than	i, x	
0010 000a	`BRV`	Branch if V	i, x	
0010 001a	`BRC`	Branch if C	i, x	
0010 010a	`CALL`	Call subroutine	i, x	
0010 011n	`NOPn`	Unary no operation trap	U	
0010 1aaa	`NOP`	Nonunary no operation trap	i	
0011 0aaa	`DECI`	Decimal input trap	d, n, s, sf, x, sx, sfx	NZV
0011 1aaa	`DECO`	Decimal output trap	i, d, n, s, sf, x, sx, sfx	
0100 0aaa	`HEXO`	Hexadecimal output trap	i, d, n, s, sf, x, sx, sfx	
0100 1aaa	`STRO`	String output trap	d, n, sf, x	
0101 0aaa	`ADDSP`	Add to stack pointer (SP)	i, d, n, s, sf, x, sx, sfx	NZVC
0101 1aaa	`SUBSP`	Subtract from stack pointer (SP)	i, d, n, s, sf, x, sx, sfx	NZVC
0110 raaa	`ADDr`	Add to r	i, d, n, s, sf, x, sx, sfx	NZVC
0111 raaa	`SUBr`	Subtract from r	i, d, n, s, sf, x, sx, sfx	NZVC
1000 raaa	`ANDr`	Bitwise AND to r	i, d, n, s, sf, x, sx, sfx	NZ
1001 raaa	`ORr`	Bitwise OR to r	i, d, n, s, sf, x, sx, sfx	NZ
1010 raaa	`CPWr`	Compare word to r	i, d, n, s, sf, x, sx, sfx	NZVC
1011 raaa	`CPBr`	Compare byte to r⟨8..15⟩	i, d, n, s, sf, x, sx, sfx	NZVC
1100 raaa	`LDWr`	Load word r from memory	i, d, n, s, sf, x, sx, sfx	NZ
1101 raaa	`LDBr`	Load byte r⟨8..15⟩ from memory	i, d, n, s, sf, x, sx, sfx	NZ
1110 raaa	`STWr`	Store word r to memory	d, n, s, sf, x, sx, sfx	
1111 raaa	`STBr`	Store byte r⟨8..15⟩ to memory	d, n, s, sf, x, sx, sfx	

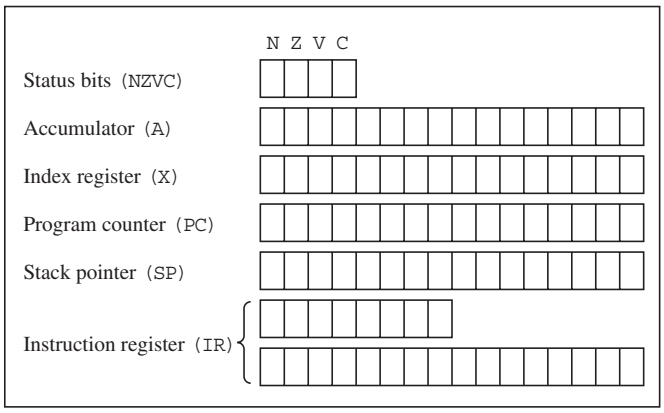
Instruction	Register transfer language specification
`STOP`	Stop execution
`RET`	$\text{PC} \leftarrow \text{Mem}[\text{SP}]$; $\text{SP} \leftarrow \text{SP}+2$
`RETTR`	$\text{NZVC} \leftarrow \text{Mem}[\text{SP}]\langle 4..7\rangle$; $\text{A} \leftarrow \text{Mem}[\text{SP}+1]$; $\text{X} \leftarrow \text{Mem}[\text{SP}+3]$; $\text{PC} \leftarrow \text{Mem}[\text{SP}+5]$; $\text{SP} \leftarrow \text{Mem}[\text{SP}+7]$
`MOVSPA`	$\text{A} \leftarrow \text{SP}$
`MOVFLGA`	$\text{A}\langle 8..11\rangle \leftarrow 0$, $\text{A}\langle 12..15\rangle \leftarrow \text{NZVC}$
`MOVAFLG`	$\text{NZVC} \leftarrow \text{A}\langle 12..15\rangle$
`NOTr`	$\text{r} \leftarrow \neg\text{r}$; $\text{N} \leftarrow \text{r}<0$, $\text{Z} \leftarrow \text{r}=0$
`NEGr`	$\text{r} \leftarrow -\text{r}$; $\text{N} \leftarrow \text{r}<0$, $\text{Z} \leftarrow \text{r}=0$, $\text{V} \leftarrow \{\textit{overflow}\}$
`ASLr`	$\text{C} \leftarrow \text{r}\langle 0\rangle$, $\text{r}\langle 0..14\rangle \leftarrow \text{r}\langle 1..15\rangle$, $\text{r}\langle 15\rangle \leftarrow 0$; $\text{N} \leftarrow \text{r}<0$, $\text{Z} \leftarrow \text{r}=0$, $\text{V} \leftarrow \{\textit{overflow}\}$
`ASRr`	$\text{C} \leftarrow \text{r}\langle 15\rangle$, $\text{r}\langle 1..15\rangle \leftarrow \text{r}\langle 0..14\rangle$; $\text{N} \leftarrow \text{r}<0$, $\text{Z} \leftarrow \text{r}=0$
`ROLr`	$\text{C} \leftarrow \text{r}\langle 0\rangle$, $\text{r}\langle 0..14\rangle \leftarrow \text{r}\langle 1..15\rangle$, $r\langle 15\rangle \leftarrow \text{C}$
`RORr`	$\text{C} \leftarrow \text{r}\langle 15\rangle$, $\text{r}\langle 1..15\rangle \leftarrow \text{r}\langle 0..14\rangle$, $r\langle 0\rangle \leftarrow \text{C}$
`BR`	$\text{PC} \leftarrow \text{Oprnd}$
`BRLE`	$\text{N}=1 \vee \text{Z}=1 \Rightarrow \text{PC} \leftarrow \text{Oprnd}$
`BRLT`	$\text{N}=1 \Rightarrow \text{PC} \leftarrow \text{Oprnd}$
`BREQ`	$\text{Z}=1 \Rightarrow \text{PC} \leftarrow \text{Oprnd}$
`BRNE`	$\text{Z}=0 \Rightarrow \text{PC} \leftarrow \text{Oprnd}$
`BRGE`	$\text{N}=0 \Rightarrow \text{PC} \leftarrow \text{Oprnd}$
`BRGT`	$\text{N}=0 \wedge \text{Z}=0 \Rightarrow \text{PC} \leftarrow \text{Oprnd}$
`BRV`	$\text{V}=1 \Rightarrow \text{PC} \leftarrow \text{Oprnd}$
`BRC`	$\text{C}=1 \Rightarrow \text{PC} \leftarrow \text{Oprnd}$
`CALL`	$\text{SP} \leftarrow \text{SP}-2$; $\text{Mem}[\text{SP}] \leftarrow \text{PC}$; $\text{PC} \leftarrow \text{Oprnd}$
`NOPn`	Trap: Unary no operation
`NOP`	Trap: Nonunary no operation
`DECI`	Trap: $\text{Oprnd} \leftarrow \{\textit{decimal input}\}$
`DECO`	Trap: $\{\textit{decimal output}\} \leftarrow \text{Oprnd}$
`HEXO`	Trap: $\{\textit{hexadecimal output}\} \leftarrow \text{Oprnd}$
`STRO`	Trap: $\{\textit{string output}\} \leftarrow \text{Oprnd}$
`ADDSP`	$\text{SP} \leftarrow \text{SP}+\text{Oprnd}$
`SUBSP`	$\text{SP} \leftarrow \text{SP}-\text{Oprnd}$
`ADDr`	$\text{r} \leftarrow \text{r}+\text{Oprnd}$; $\text{N} \leftarrow \text{r}<0$, $\text{Z} \leftarrow \text{r}=0$, $\text{V} \leftarrow \{\textit{overflow}\}$, $\text{C} \leftarrow \{\textit{carry}\}$
`SUBr`	$\text{r} \leftarrow \text{r}-\text{Oprnd}$; $\text{N} \leftarrow \text{r}<0$, $\text{Z} \leftarrow \text{r}=0$, $\text{V} \leftarrow \{\textit{overflow}\}$, $\text{C} \leftarrow \{\textit{carry}\}$
`ANDr`	$\text{r} \leftarrow \text{r} \wedge \text{Oprnd}$; $\text{N} \leftarrow \text{r}<0$, $\text{Z} \leftarrow \text{r}=0$
`ORr`	$\text{r} \leftarrow \text{r} \vee \text{Oprnd}$; $\text{N} \leftarrow \text{r}<0$, $\text{Z} \leftarrow \text{r}=0$
`CPWr`	$\text{T} \leftarrow \text{r}-\text{Oprnd}$; $\text{N} \leftarrow \text{T}<0$, $\text{Z} \leftarrow \text{T}=0$, $\text{V} \leftarrow \{\textit{overflow}\}$, $\text{C} \leftarrow \{\textit{carry}\}$; $\text{N} \leftarrow \text{N} \oplus \text{V}$
`CPBr`	$\text{T} \leftarrow \text{r}\langle 8..15\rangle - \text{byte Oprnd}$; $\text{N} \leftarrow \text{T}<0$, $\text{Z} \leftarrow \text{T}=0$, $\text{V} \leftarrow 0$, $\text{C} \leftarrow 0$
`LDWr`	$\text{r} \leftarrow \text{Oprnd}$; $\text{N} \leftarrow \text{r}<0$, $\text{Z} \leftarrow \text{r}=0$
`LDBr`	$\text{r}\langle 8..15\rangle \leftarrow \text{byte Oprnd}$; $\text{N} \leftarrow 0$, $\text{Z} \leftarrow \text{r}\langle 8..15\rangle = 0$
`STWr`	$\text{Oprnd} \leftarrow \text{r}$
`STBr`	$\text{byte Oprnd} \leftarrow \text{r}\langle 8..15\rangle$
Trap	$\text{T} \leftarrow \text{Mem}[\text{FFF6}]$; $\text{Mem}[\text{T}-1] \leftarrow \text{IR}\langle 0..7\rangle$; $\text{Mem}[\text{T}-3] \leftarrow \text{SP}$; $\text{Mem}[\text{T}-5] \leftarrow \text{PC}$; $\text{Mem}[\text{T}-7] \leftarrow \text{X}$; $\text{Mem}[\text{T}-9] \leftarrow \text{A}$; $\text{Mem}[\text{T}-10]\langle 4..7\rangle \leftarrow \text{NZVC}$; $\text{SP} \leftarrow \text{T}-10$; $\text{PC} \leftarrow \text{Mem}[\text{FFFE}]$

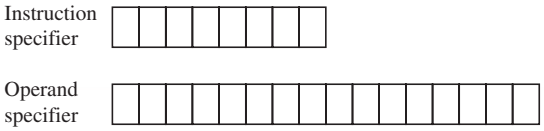
Char	Bin	Hex	Char	Bin	Hex	Char	Bin	Hex	Char	Bin	Hex
NUL	000 0000	00	SP	010 0000	20	@	100 0000	40	`	110 0000	60
SOH	000 0001	01	!	010 0001	21	A	100 0001	41	a	110 0001	61
STX	000 0010	02	"	010 0010	22	B	100 0010	42	b	110 0010	62
ETX	000 0011	03	#	010 0011	23	C	100 0011	43	c	110 0011	63
EOT	000 0100	04	$	010 0100	24	D	100 0100	44	d	110 0100	64
ENQ	000 0101	05	%	010 0101	25	E	100 0101	45	e	110 0101	65
ACK	000 0110	06	&	010 0110	26	F	100 0110	46	f	110 0110	66
BEL	000 0111	07	'	010 0111	27	G	100 0111	47	g	110 0111	67
BS	000 1000	08	(	010 1000	28	H	100 1000	48	h	110 1000	68
HT	000 1001	09	)	010 1001	29	I	100 1001	49	i	110 1001	69
LF	000 1010	0A	*	010 1010	2A	J	100 1010	4A	j	110 1010	6A
VT	000 1011	0B	+	010 1011	2B	K	100 1011	4B	k	110 1011	6B
FF	000 1100	0C	,	010 1100	2C	L	100 1100	4C	l	110 1100	6C
CR	000 1101	0D	–	010 1101	2D	M	100 1101	4D	m	110 1101	6D
SO	000 1110	0E	.	010 1110	2E	N	100 1110	4E	n	110 1110	6E
SI	000 1111	0F	/	010 1111	2F	O	100 1111	4F	o	110 1111	6F
DLE	001 0000	10	0	011 0000	30	P	101 0000	50	p	111 0000	70
DC1	001 0001	11	1	011 0001	31	Q	101 0001	51	q	111 0001	71
DC2	001 0010	12	2	011 0010	32	R	101 0010	52	r	111 0010	72
DC3	001 0011	13	3	011 0011	33	S	101 0011	53	s	111 0011	73
DC4	001 0100	14	4	011 0100	34	T	101 0100	54	t	111 0100	74
NAK	001 0101	15	5	011 0101	35	U	101 0101	55	u	111 0101	75
SYN	001 0110	16	6	011 0110	36	V	101 0110	56	v	111 0110	76
ETB	001 0111	17	7	011 0111	37	W	101 0111	57	w	111 0111	77
CAN	001 1000	18	8	011 1000	38	X	101 1000	58	x	111 1000	78
EM	001 1001	19	9	011 1001	39	Y	101 1001	59	y	111 1001	79
SUB	001 1010	1A	:	011 1010	3A	Z	101 1010	5A	z	111 1010	7A
ESC	001 1011	1B	;	011 1011	3B	[	101 1011	5B	{	111 1011	7B
FS	001 1100	1C	<	011 1100	3C	\	101 1100	5C	|	111 1100	7C
GS	001 1101	1D	=	011 1101	3D	]	101 1101	5D	}	111 1101	7D
RS	001 1110	1E	>	011 1110	3E	^	101 1110	5E	~	111 1110	7E
US	001 1111	1F	?	011 1111	3F	_	101 1111	5F	DEL	111 1111	7F

Abbreviations for Control Characters

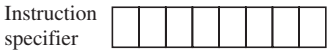
NUL	null, or all zeros	**FF**	form feed	**CAN**	cancel
SOH	start of heading	**CR**	carriage return	**EM**	end of medium
STX	start of text	**SO**	shift out	**SUB**	substitute
ETX	end of text	**SI**	shift in	**ESC**	escape
EOT	end of transmission	**DLE**	data link escape	**FS**	file separator
ENQ	enquiry	**DC1**	device control 1	**GS**	group separator
ACK	acknowledge	**DC2**	device control 2	**RS**	record separator
BEL	bell	**DC3**	device control 3	**US**	unit separator
BS	backspace	**DC4**	device control 4	**SP**	space
HT	horizontal tabulation	**NAK**	negative acknowledge	**DEL**	delete
LF	line feed	**SYN**	synchronous idle		
VT	vertical tabulation	**ETB**	end of transmission block		

Central processing unit (CPU)





(a) The two parts of a nonunary instruction.



(b) A unary instruction.

aaa	Addressing Mode
000	Immediate
001	Direct
010	Indirect
011	Stack-relative
100	Stack-relative deferred
101	Indexed
110	Stack-indexed
111	Stack-deferred indexed

(a) The addressing-aaa field.

a	Addressing Mode
0	Immediate
1	Indexed

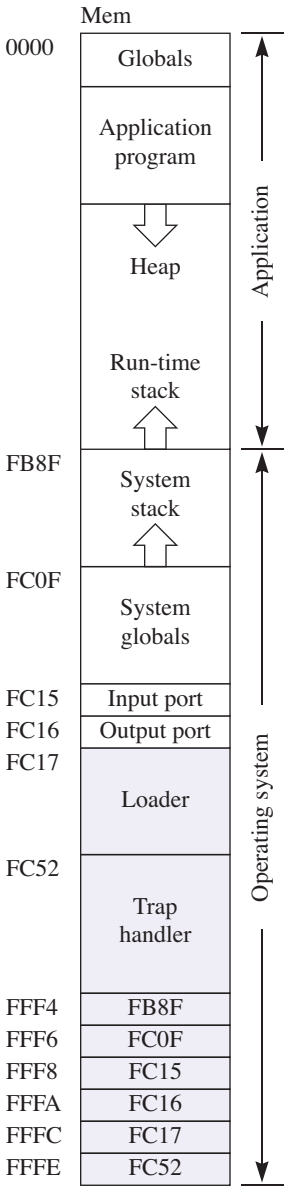
(b) The addressing-a field.

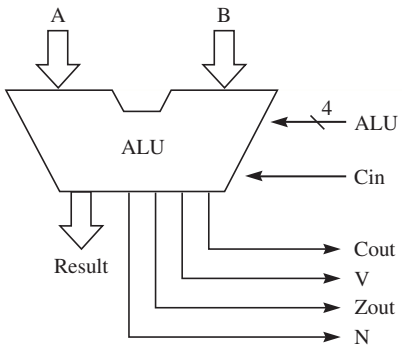
r	Register
0	Accumulator, A
1	Index register, X

(c) The register-r field.

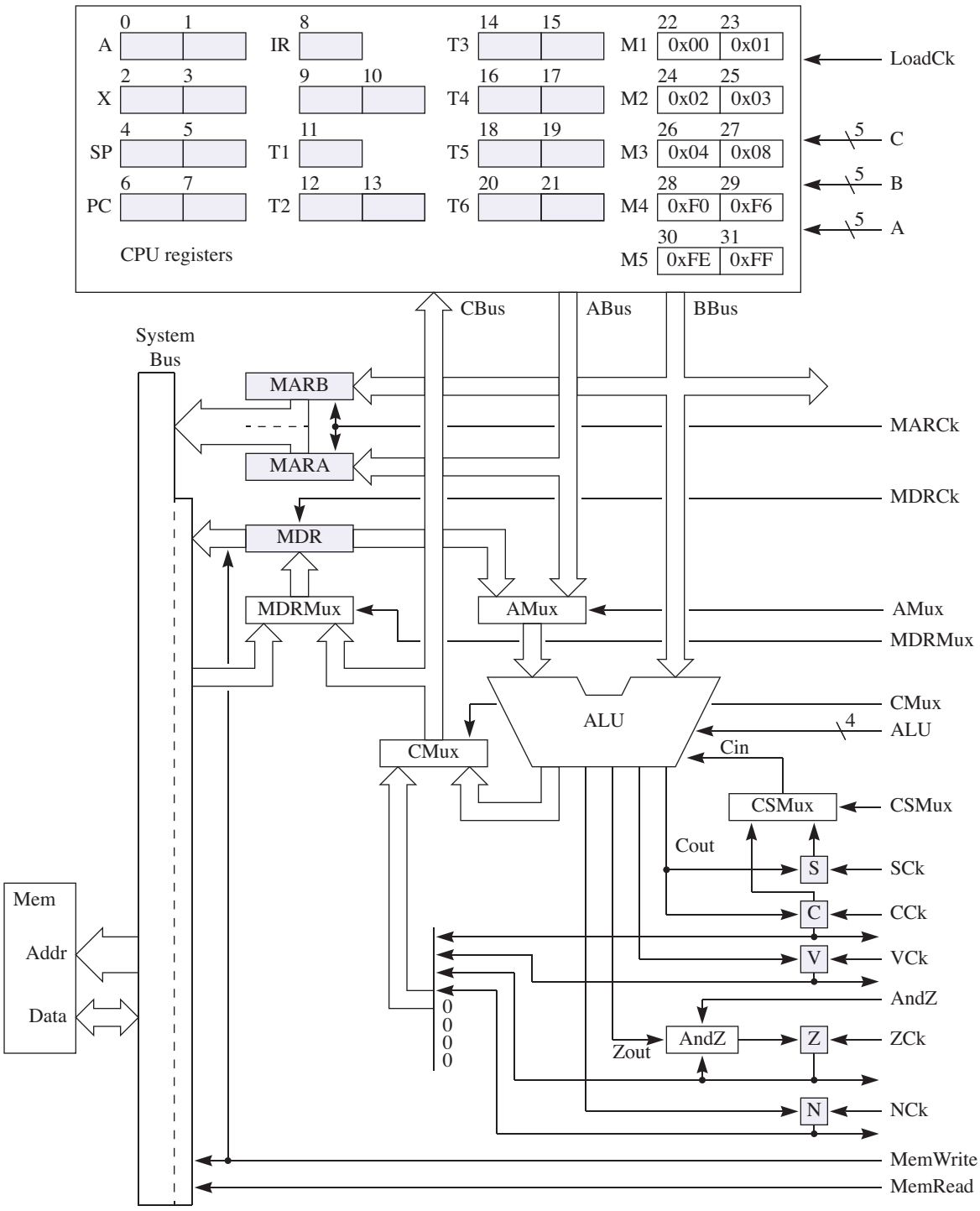
Addressing Mode	aaa	Letters	Operand
Immediate	000	i	OprndSpec
Direct	001	d	Mem[OprndSpec]
Indirect	010	n	Mem[Mem[OprndSpec]]
Stack-relative	011	s	Mem[SP + OprndSpec]
Stack-relative deferred	100	sf	Mem[Mem[SP + OprndSpec]]
Indexed	101	x	Mem[OprndSpec + X]
Stack-indexed	110	sx	Mem[SP + OprndSpec + X]
Stack-deferred indexed	111	sfx	Mem[Mem[SP + OprndSpec] + X]

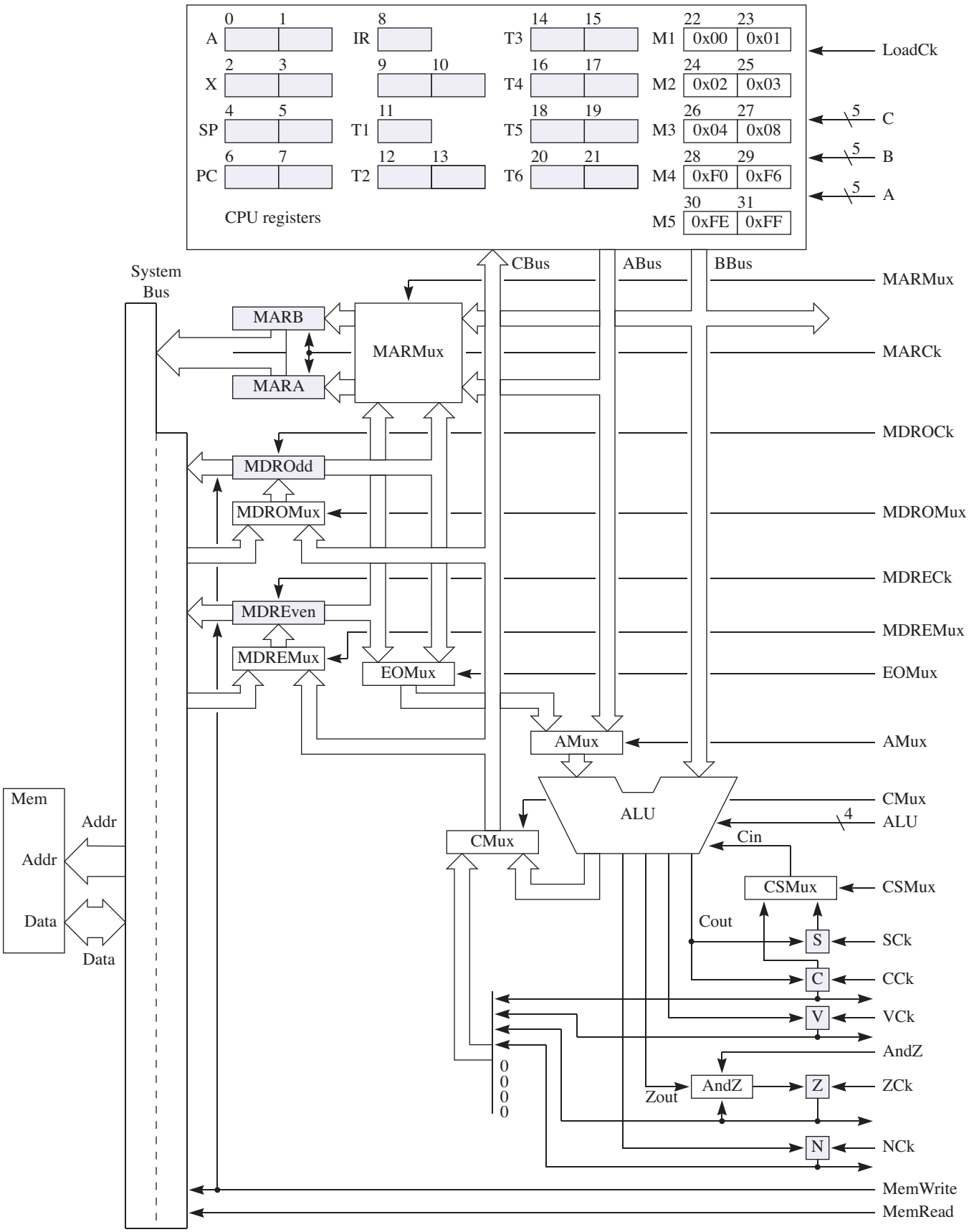
The shaded portion is ROM.





ALU Control			Status Bits			
(bin)	(dec)	Result	N	Zout	V	Cout
0000	0	A	N	Z	0	0
0001	1	A plus B	N	Z	V	C
0010	2	A plus B plus Cin	N	Z	V	C
0011	3	A plus $\overline{B}$ plus 1	N	Z	V	C
0100	4	A plus $\overline{B}$ plus Cin	N	Z	V	C
0101	5	A · B	N	Z	0	0
0110	6	$\overline{A \cdot B}$	N	Z	0	0
0111	7	A + B	N	Z	0	0
1000	8	$\overline{A + B}$	N	Z	0	0
1001	9	A $\oplus$ B	N	Z	0	0
1010	10	$\overline{A}$	N	Z	0	0
1011	11	ASL A	N	Z	V	C
1100	12	ROL A	N	Z	V	C
1101	13	ASR A	N	Z	0	C
1110	14	ROR A	N	Z	0	C
1111	15	0	A<4>	A<5>	A<6>	A<7>


0 1
A
2 3
X
4 5
SP
6 7
PC
CPU registers
8
IR
9 10
11
T1
12 13
T2
14 15
T3
16 17
T4
18 19
T5
20 21
T6
22 23
M1 0x00 0x01
24 25
M2 0x02 0x03
26 27
M3 0x04 0x08
28 29
M4 0xF0 0xF6
30 31
M5 0xFE 0xFF
LoadCk
5
C
5
B
5
A
CBus
ABus
BBus
System Bus
MARB
MARA
MARCk
MDRCk
MDR
MDRMux
AMux
AMux
MDRMux
ALU
CMux
ALU
4
Cin
CMux
CSMux
CSMux
Cout
S
SCk
C
CCk
V
VCk
AndZ
Mem
Addr
Data
0
0
0
0
AndZ
Zout
Z
ZCk
N
NCk
MemWrite
MemRead


0 1
A
2 3
X
4 5
SP
6 7
PC
CPU registers
8
IR
9 10
11
T1
12 13
T2
14 15
T3
16 17
T4
18 19
T5
20 21
T6
22 23
M1 0x00 0x01
24 25
M2 0x02 0x03
26 27
M3 0x04 0x08
28 29
M4 0xF0 0xF6
30 31
M5 0xFE 0xFF
LoadCk
5
C
5
B
5
A
System Bus
CBus
ABus
BBus
MARMux
MARCk
MDROCk
MDROMux
MDRECk
MDREMux
EOMux
AMux
CMux
ALU
4
Cin
CSMux
Cout
SCk
CCk
VCk
AndZ
ZCk
NCk
MemWrite
MemRead
MARB
MARA
MARMux
MDROdd
MDROMux
MDREven
MDREMux
EOMux
AMux
ALU
CMux
CSMux
S
C
V
Z
N
AndZ
Zout
0
0
0
0
Mem
Addr
Addr
Data
Data

The memory read bus protocol

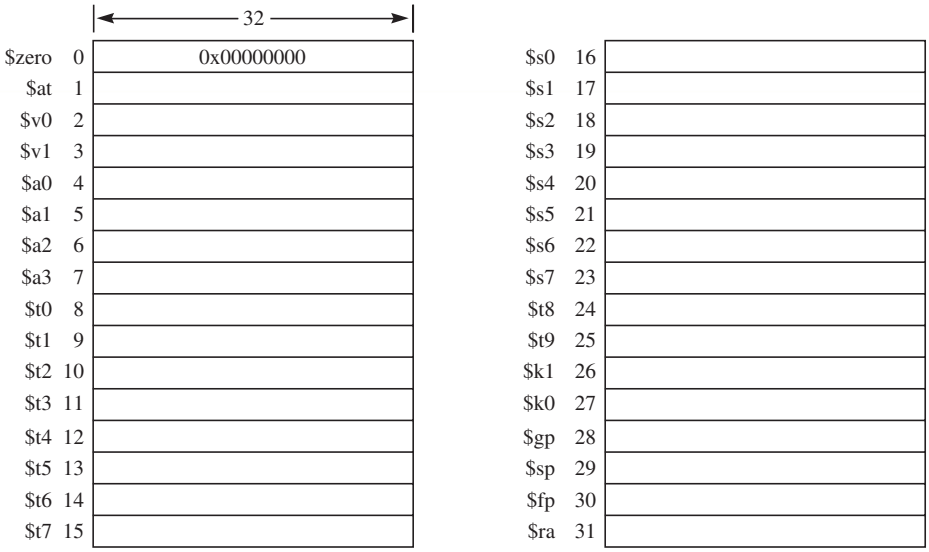
The bus protocol for a memory read over the Pep/9 system bus requires three consecutive cycles, with MemRead asserted on each cycle. The read operation must adhere to the following specification:

- You must clock the address into the MAR before the first MemRead cycle.
- You must clock the data into the MDR from the system bus on or before the third MemRead cycle.
- On the third MemRead cycle, you cannot clock a new value into MAR in anticipation of a following memory operation.

The memory write bus protocol

The bus protocol for a memory write requires three consecutive cycles, with MemWrite asserted on each cycle. The write operation must adhere to the following specification:

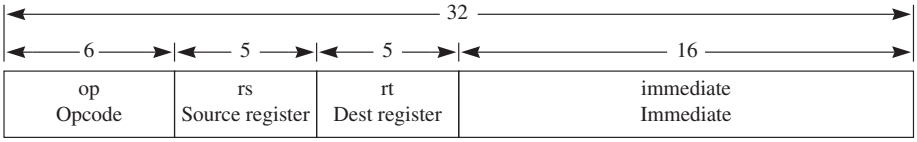
- You must clock the address into the MAR before the first MemWrite cycle.
- On the first or second MemWrite cycle, you can clock the data to be written into the MDR.
- On the third MemWrite cycle, you can clock a new data value into the MDR in anticipation of a following memory write. However, you cannot clock a new address value into the MAR in anticipation of a following memory operation.



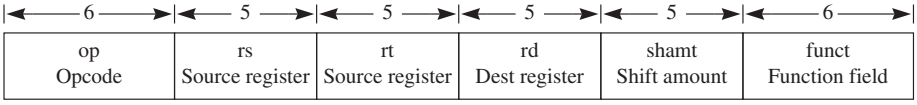
(a) MIPS registers.

(b) Pep/9 registers.

Addressing Mode	Instruction Type	Operands		
		Destination	Source	Source
Immediate	I-type	Reg[rt]	Reg[rs]	SE(im)
Register	R-type	Reg[rd]	Reg[rs]	Reg[rt]
Base with load	I-type	Reg[rt]	Mem[Reg[rb] + SE(im)]	
Base with store	I-type	Mem[Reg[rb] + SE(im)]	Reg[rt]	
PC-relative	I-type	PC	PC + 4	SE(im × 4)
Pseudodirect	J-type	PC	(PC + 4)⟨0..3⟩ : (ta × 4)	

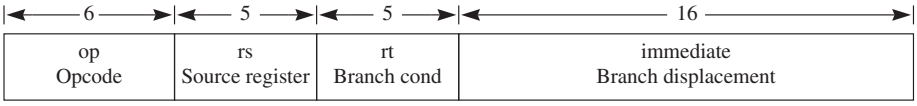


(a) Immediate addressing with the I-type instruction.

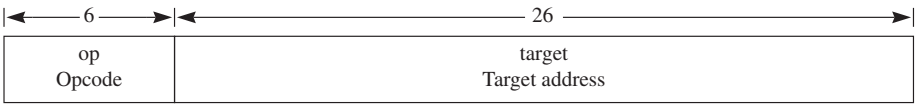


(b) Register addressing with the R-type instruction.

(c) Base addressing with the I-type instruction.

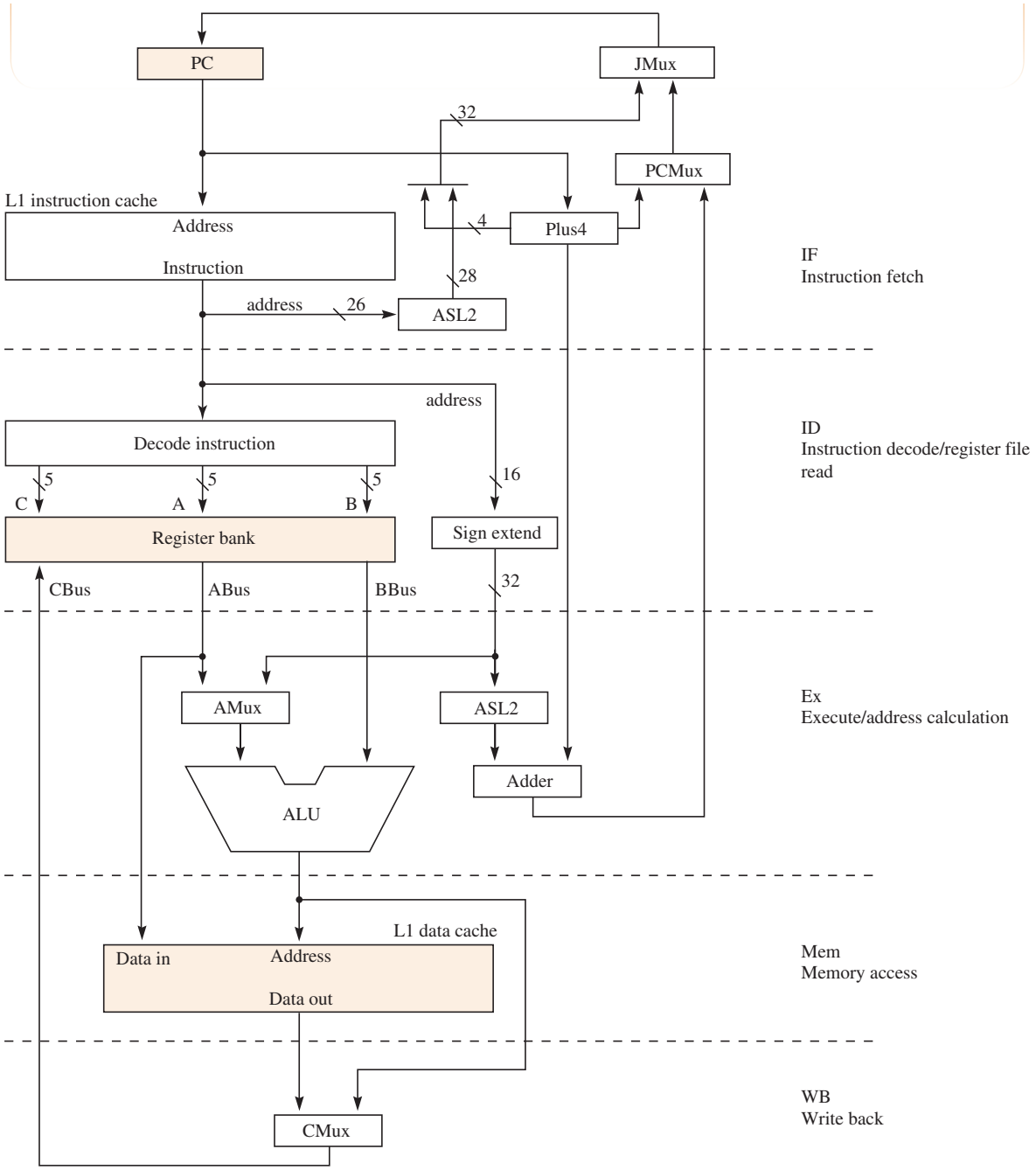


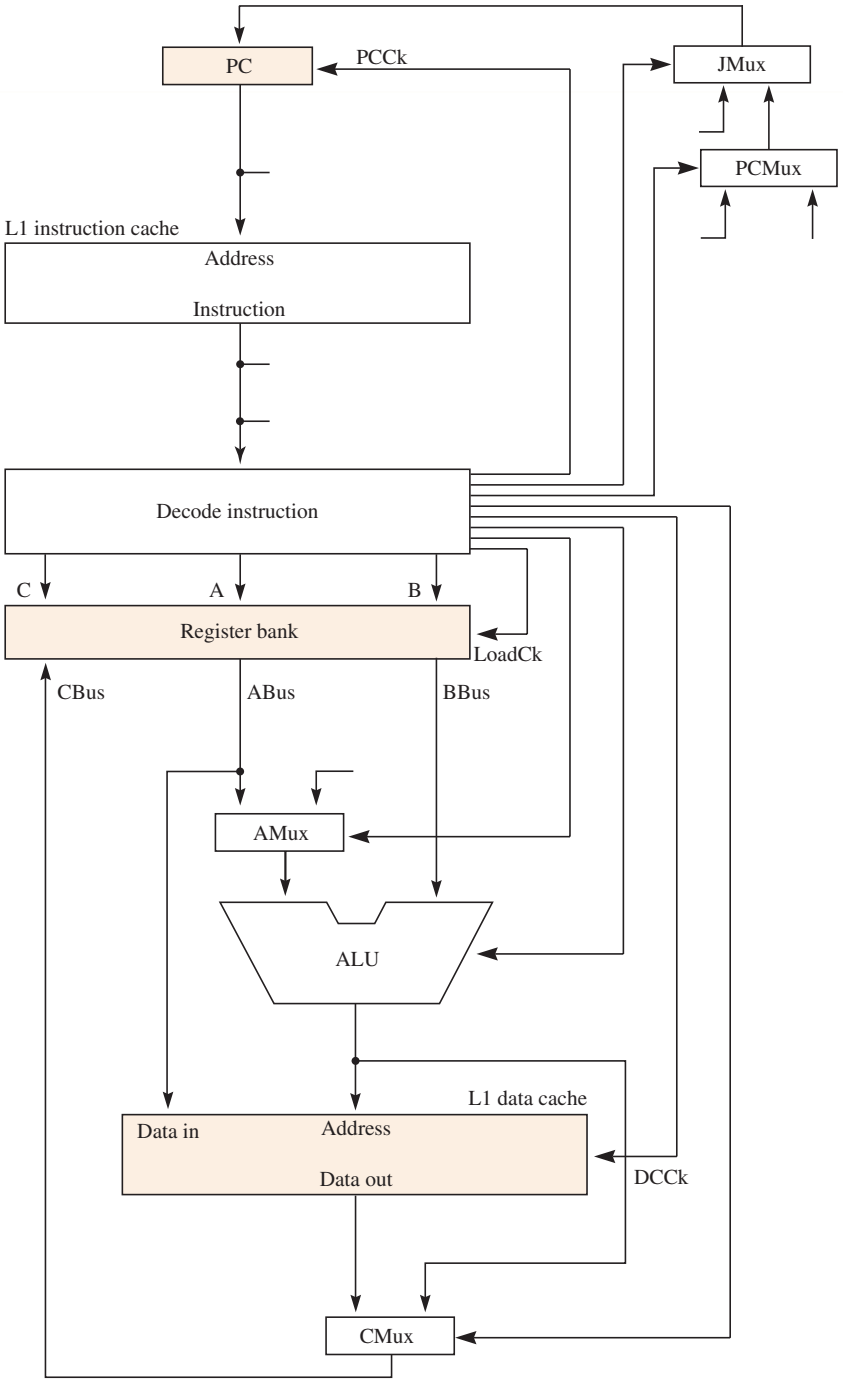
(d) PC-relative addressing with the I-type instruction.

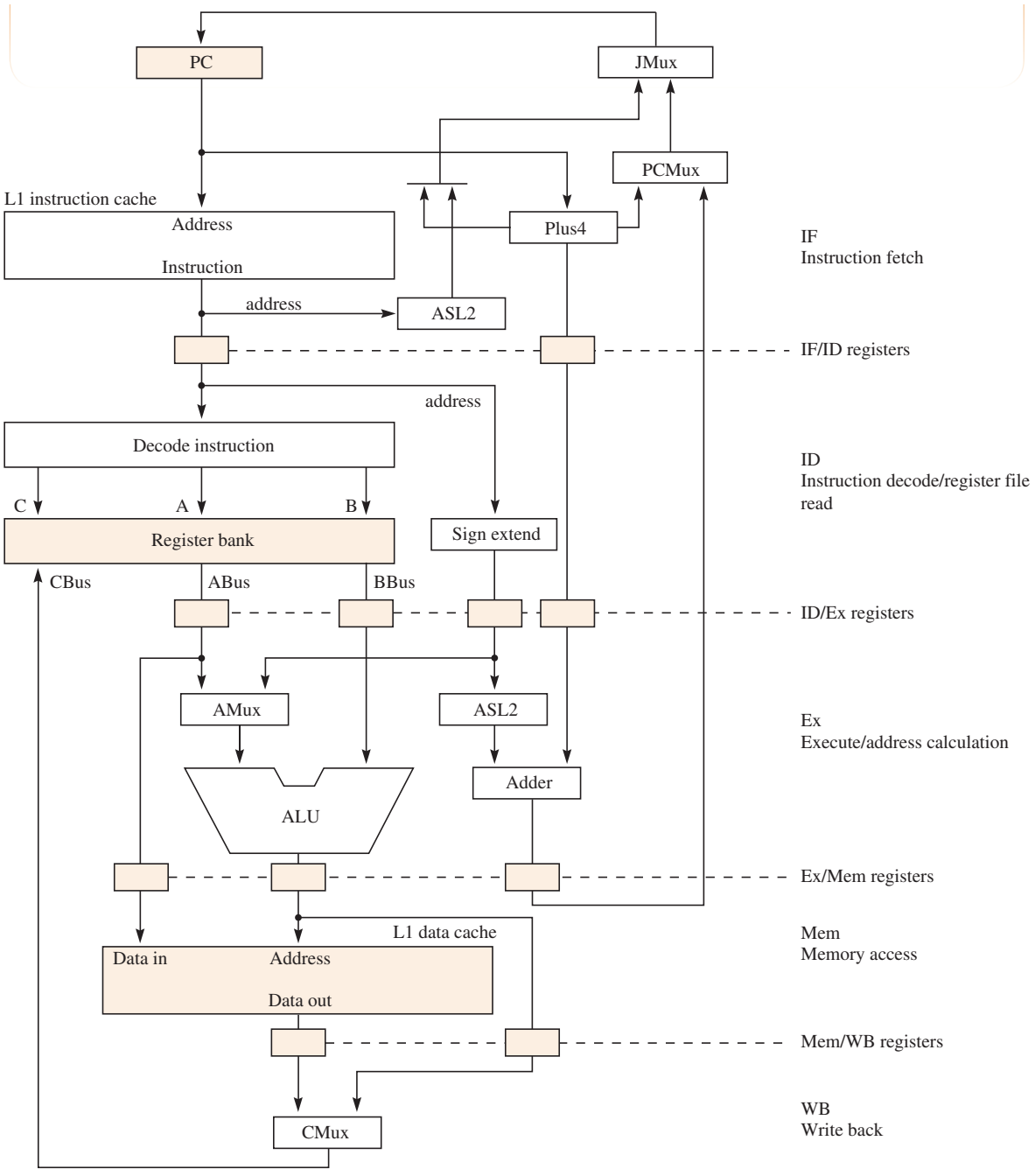


(e) Pseudodirect addressing with the J-type instruction.

Mnemonic	Meaning	Binary Instruction Encoding							
add	Add	0000	00ss	ssst	tttt	dddd	d000	0010	0000
addi	Add immediate	0010	00ss	sssd	dddd	iiii	iiii	iiii	iiii
sub	Subtract	0000	00ss	ssst	tttt	dddd	d000	0010	0010
and	Bitwise AND	0000	00ss	ssst	tttt	dddd	d000	0010	0100
andi	Bitwise AND immediate	0011	00ss	sssd	dddd	iiii	iiii	iiii	iiii
or	Bitwise OR	0000	00ss	ssst	tttt	dddd	d000	0010	0101
ori	Bitwise OR immediate	0011	01ss	sssd	dddd	iiii	iiii	iiii	iiii
sll	Shift left logical	0000	0000	000t	tttt	dddd	dhhh	hh00	0000
sra	Shift right arithmetic	0000	0000	000t	tttt	dddd	dhhh	hh00	0011
srl	Shift right logical	0000	0000	000t	tttt	dddd	dhhh	hh00	0010
lb	Load byte	1000	00bb	bbbd	dddd	aaaa	aaaa	aaaa	aaaa
lw	Load word	1000	11bb	bbbd	dddd	aaaa	aaaa	aaaa	aaaa
lui	Load upper immediate	0011	1100	000d	dddd	iiii	iiii	iiii	iiii
sb	Store byte	1010	00bb	bbbt	tttt	aaaa	aaaa	aaaa	aaaa
sw	Store word	1010	11bb	bbbt	tttt	aaaa	aaaa	aaaa	aaaa
beq	Branch if equal to	0001	00ss	ssst	tttt	aaaa	aaaa	aaaa	aaaa
bgez	Branch if greater than or equal to zero	0000	01ss	sss0	0001	aaaa	aaaa	aaaa	aaaa
bgtz	Branch if greater than zero	0001	11ss	sss0	0000	aaaa	aaaa	aaaa	aaaa
blez	Branch if less than or equal to zero	0001	10ss	sss0	0000	aaaa	aaaa	aaaa	aaaa
bltz	Branch if less than zero	0000	01ss	sss0	0000	aaaa	aaaa	aaaa	aaaa
bne	Branch if not equal to	0001	01ss	ssst	tttt	aaaa	aaaa	aaaa	aaaa
j	Jump	0000	10aa	aaaa	aaaa	aaaa	aaaa	aaaa	aaaa
jr	Jump register	0000	00bb	bbb0	0000	0000	0000	0000	1000


PC
JMux
32
PCMux
L1 instruction cache
Address
Instruction
4
Plus4
28
address
26
ASL2
IF
Instruction fetch
address
Decode instruction
5
5
5
16
C
A
B
Register bank
Sign extend
CBus
ABus
BBus
32
ID
Instruction decode/register file read
AMux
ASL2
ALU
Adder
Ex
Execute/address calculation
L1 data cache
Data in
Address
Data out
Mem
Memory access
CMux
WB
Write back


PC
PCCk
JMux
PCMux
L1 instruction cache
Address
Instruction
Decode instruction
C
A
B
Register bank
LoadCk
CBus
ABus
BBus
AMux
ALU
L1 data cache
Data in
Address
Data out
DCCk
CMux


PC
JMux
PCMux
L1 instruction cache
Address
Instruction
Plus4
address
ASL2
IF
Instruction fetch
IF/ID registers
address
Decode instruction
ID
Instruction decode/register file read
C
A
B
Register bank
Sign extend
CBus
ABus
BBus
ID/Ex registers
AMux
ASL2
Ex
Execute/address calculation
Adder
ALU
Ex/Mem registers
L1 data cache
Data in
Address
Data out
Mem
Memory access
Mem/WB registers
CMux
WB
Write back