C-3PU

Microprocessor

Ognjen
Glamočanin
EE 15/2013

Assembler instructions

1. Memory isntructions:

STORE	LOAD	MOVE
STR var, R_x	LDR R_x, var	MOV R_x, R_y
STR R_x, [Ry + const]	LDR R_x, #const	
STR R_x, [Ry += const]	LDR R_x, [Ry + const]	
STR R_x, [Ry], #const	LDR R_x, [Ry += const]	
STR R_x, [Ry + R_z]	LDR R_x, [Ry], #const	
STR R_x, [Ry += R_z]	LDR R_x, [Ry + R_z]	
STR R_x, [Ry], R_z	LDR R_x, [Ry += R_z]	
	LDR R_x, [Ry], R_z	

INSTRUCTION	DESCRIPTION
MOV R_x, R_y	Move the content from R_y to R_s
STR var, R_x	Store the content from R_x in variable var
STR R_x, [Ry + const]	Store the content from R_x to the address from R_y added with const
STR R_x, [Ry += const]	Store the content from R_x to the address from R_y added with const, R_y is incremented by const
STR R_x, [Ry], #const	Store the content from R_x to the address from R_y, R_y is incremented by const
STR R_x, [Ry + R_z]	Store the content from R_x to the address from R_y added with R_z
STR R_x, [Ry += R_z]	Store the content from R_x to the address from R_y added with R_z, R_y is incremented by R_z
STR R_x, [Ry], R_z	Store the content from R_x to the address from R_y, R_y is incremented by R_z
LDR R_x, var	Load the content from var to R_x
LDR R_x, #const	Load the constant into R_x
LDR R_x, [Ry + const]	Load the content to R_x from the address from R_y added with const
LDR R_x, [Ry += const]	Load the content to R_x from the address from R_y added with const, R_y is incremented by const
LDR R_x, [Ry], #const	Load the content to R_x from the address from R_y, R_y is incremented by const
LDR R_x, [Ry + R_z]	Load the content to R_x from the address from R_y added with R_z
LDR R_x, [Ry += R_z]	Load the content to R_x from the address from R_y added with R_z, R_y is incremented by R_z
LDR R_x, [Ry], R_z	Load the content to R_x from the address from R_y, R_y is incremented by R_z

2. Data processing instructions:

ARITMETRICAL	LOGICAL
ADD R_x, R_y, R_z	AND R_x, R_y, R_z
ADD R_x, R_y, #const	AND R_x, R_y, #const
SUB R_x, R_y, R_z	OR R_x, R_y, R_z
SUB R_x, R_y, #const	OR R_x, R_y, #const
CLR R_x	XOR R_x, R_y, R_z
	XOR R_x, R_y, #const
	NOT R_x
	SHR R_x
	SHL R_x

INSTRUCTION	DESCRIPTION
ADD R_x, R_y, R_z	Store into R_x the result of the sum of the contents of R_y and R_z
ADD R_x, R_y, #const	Store into R_x the result of the sum of the content of R_y and const
SUB R_x, R_y, R_z	Store into R_x the result of the subtraction of the contents of R_y and R_z
SUB R_x, R_y, #const	Store into R_x the result of the subtraction of the content of R_y and const
CLR R_x	Store 0x00000000 into R_x
AND R_x, R_y, R_z	Store into R_x the result of the logical opperation AND of the contents of R_y and R_z
AND R_x, R_y, #const	Store into R_x the result of the logical opperation AND of the content of R_y and const
OR R_x, R_y, R_z	Store into R_x the result of the logical opperation OR of the contents of R_y and R_z
OR R_x, R_y, #const	Store into R_x the result of the logical opperation OR of the content of R_y and const
XOR R_x, R_y, R_z	Store into R_x the result of the logical opperation XOR of the contents of R_y and R_z
XOR R_x, R_y, #const	Store into R_x the result of the logical opperation XOR of the content of R_y and const
NOT R_x	Store into R_x the result of the logical opperation NOT of the content of R_x
SHR R_x	Store into R_x the result of the logical opperation SHIFT RIGHT of the content of R_x
SHL R_x	Store into R_x the result of the logical opperation SHIFT LEFT of the content of R_x

3. Branch instructions:

JUMP
JMP label
JZ label
JC label

INSTRUCTION	DESCRIPTION
JMP label	Jump to label
JZ label	Jump to label if the result of the previous instruction is zero
JC label	Jump to label if the result of the previous instruction had generated a carry bit

4. Special instructions:

SPECIAL
NOP
END

INSTRUCTION	DESCRIPTION
NOP	No operation
END	End of program

Instruction formats

1. Memory isntructions:

LDR, STR:

31 - 27	26 - 23	22 - 18	18 - 0
OPCODE	DESTINATION	BASE	OFSET
5-bit	4-bit	4-bit	19-bit

18 - 15	14 - 0
REGISTER	x
4-bit	15-bit

18 - 0
19-bit CONST
19-bit

MOV:

31 - 27	26 - 23	22 - 18	18 - 0
OPCODE	DESTINATION	SOURCE	x
5-bit	4-bit	4-bit	19-bit

2. Data processing instructions:

- two operands:

31 - 27	26 - 23	22 - 18	18 - 0
OPCODE	DESTINATION	SOURCE 1	SOURCE 2
5-bit	4-bit	4-bit	19-bit

- one operand:

31 - 27	26 - 23	22 - 0
OPCODE	DEST/SOURCE	x
5-bit	4-bit	23-bit

3. Branch instructions:

31 - 27	26 - 0
OPCODE	ADDRESS
5-bit	27-bit

4. Special instructions:

31 - 27	26 - 0
OPCODE	x
5-bit	27-bit

ISA

INSTRUCTION	DESCRIPTION	OPCODE
LDR R_x, [R_y + const]	**R_x = mem[R_y + const]**	**00000**
CLOCK	RTN	CONTROL SIGNALS
T0	IR <- IM [PC]	IR_{IN}, MOP=1
T1	XREG0 <- $RF[22-19]^{Ry}$ PC <- PC + 1	$XREG0_{IN}$ MA1=1, MA2=10, ALU=ADD, MPC=0, PC_{IN}
T2	R <- XREG0 + exp	R_{IN}, cexp=0, ALU=ADD, MA1=0, MA2=01, MR=0
T3	$RF[26-23]^{Rx}$ <- DM[R]	MRF3=00, RFW3=1, MRF2=0

INSTRUCTION	DESCRIPTION	OPCODE
LDR R_x, [R_y += const]	**R_x = mem[R_y + const], R_y = R_y + const**	**00001**
CLOCK	RTN	CONTROL SIGNALS
T0	IR <- IM [PC]	IR_{IN}, MOP=1
T1	XREG0 <- $RF[22-19]^{Ry}$ PC <- PC + 1	$XREG0_{IN}$ MA1=1, MA2=10, ALU=ADD, MPC=0, PC_{IN}
T2	R <- XREG0 + exp $RF[22-19]^{Ry}$ <- XREG0 + exp	R_{IN}, cexp=0, ALU=ADD, MA1=0, MA2=01, MR=0 MRF3=01, RFW3=1, MRF2=1
T3	$RF[26-23]^{Rx}$ <- DM[R]	MRF3=00, RFW3=1, MRF2=0

INSTRUCTION	DESCRIPTION	OPCODE
LDR R_x, [R_y], #const	**R_x = mem[R_y], R_y = R_y + const**	**00010**
CLOCK	RTN	CONTROL SIGNALS
T0	IR <- IM [PC]	IR_{IN}, MOP=1
T1	XREG0 <- $RF[22-19]^{Ry}$ PC <- PC + 1	$XREG0_{IN}$ MA1=1, MA2=10, ALU=ADD, MPC=0, PC_{IN}
T2	R <- XREG0 $RF[22-19]^{Ry}$ <- XREG0 + exp	R_{IN}, MR1=1 cexp=0, ALU=ADD, MA1=0, MA2=01, MRF3=01, RFW3=1, MRF2=1
T3	$RF[26-23]^{Rx}$ <- DM[R]	MRF3=00, RFW3=1, MRF2=0

INSTRUCTION	DESCRIPTION	OPCODE
LDR R_x, [R_y + R_z]	**R_x = mem[R_y + R_z]**	**00011**
CLOCK	RTN	CONTROL SIGNALS
T0	IR <- IM [PC]	IR_{IN}, MOP=1
T1	XREG0 <- $RF[22-19]^{Ry}$ XREG1 <- $RF[18-15]^{Rz}$ PC <- PC + 1	$XREG0_{IN}$ $XREG1_{IN}$, MRF1=0 MA1=1, MA2=10, ALU=ADD, MPC=0, PC_{IN}
T2	R <- XREG0 + XREG1	R_{IN}, ALU=ADD, MA1=0, MA2=00, MR=0
T3	$RF[26-23]^{Rx}$ <- DM[R]	MRF3=00, RFW3=1, MRF2=0

INSTRUCTION	DESCRIPTION	OPCODE
LDR R_x, [R_y += R_z]	**R_x = mem[R_y + R_z], R_y = R_y + R_z**	**00100**

CLOCK	RTN	CONTROL SIGNALS
T0	IR <- IM [PC]	IR$_{IN}$, MOP=1
T1	XREG0 <- RF[22-19]^{Ry} XREG1 <- RF[18-15]^{Rz} PC <- PC + 1	XREG0$_{IN}$ XREG1$_{IN}$, MRF1=0 MA1=1, MA2=10, ALU=ADD, MPC=0, PC$_{IN}$
T2	R <- XREG0 + XREG1 RF[22-19]^{Ry} <- XREG0 + XREG1	R$_{IN}$, ALU=ADD, MA1=0, MA2=00, MR=0 MRF3=01, RFW3=1, MRF2=1
T3	RF[26-23]^{Rx} <- DM[R]	MRF3=00, RFW3=1, MRF2=0

INSTRUCTION	DESCRIPTION	OPCODE
LDR R_x, [R_y], R_z	**R_x = mem[R_y], R_y = R_y + R_z**	**00101**

CLOCK	RTN	CONTROL SIGNALS
T0	IR <- IM [PC]	IR$_{IN}$, MOP=1
T1	XREG0 <- RF[22-19]^{Ry} XREG1 <- RF[18-15]^{Rz} PC <- PC + 1	XREG0$_{IN}$ XREG1$_{IN}$, MRF1=0 MA1=1, MA2=10, ALU=ADD, MPC=0, PC$_{IN}$
T2	R <- XREG0 RF[22-19]^{Ry} <- XREG0 + XREG1	R$_{IN}$, MR1=1 ALU=ADD, MA1=0, MA2=00, MRF3=01, RFW3=1, MRF2=1
T3	RF[26-23]^{Rx} <- DM[R]	MRF3=00, RFW3=1, MRF2=0

INSTRUCTION	DESCRIPTION	OPCODE
STR R_x, [R_y + const]	**mem[R_y + const] = R_x**	**00110**

CLOCK	RTN	CONTROL SIGNALS
T0	IR <- IM [PC]	IR$_{IN}$, MOP=1
T1	XREG0 <- RF[22-19]^{Ry} PC <- PC + 1	XREG0$_{IN}$ MA1=1, MA2=10, ALU=ADD, MPC=0, PC$_{IN}$
T2	R <- XREG0 + exp	R$_{IN}$, cexp=0, ALU=ADD, MA1=0, MA2=01, MR=0
T3	DM[R] <- RF[26-23]^{Rx}	MRF1=1, DMW=1

INSTRUCTION	DESCRIPTION	OPCODE
STR R_x, [R_y += const]	**mem[R_y + const] = R_x, R_y = R_y + const**	**00111**

CLOCK	RTN	CONTROL SIGNALS
T0	IR <- IM [PC]	IR$_{IN}$, MOP=1
T1	XREG0 <- RF[22-19]^{Ry} PC <- PC + 1	XREG0$_{IN}$ MA1=1, MA2=10, ALU=ADD, MPC=0, PC$_{IN}$
T2	R <- XREG0 + exp RF[22-19]^{Ry} <- XREG0 + exp	R$_{IN}$, cexp=0, ALU=ADD, MA1=0, MA2=01, MR=0 MRF3=01, RFW3=1, MRF2=1
T3	DM[R] <- RF[26-23]^{Rx}	MRF1=1, DMW=1

INSTRUCTION	**DESCRIPTION**	**OPCODE**
STR R_x, [R_y], #const	**mem[R_y] = R_x, R_y = R_y + const**	**01000**
CLOCK	RTN	CONTROL SIGNALS
T0	IR <- IM [PC]	IR_{IN}, MOP=1
T1	XREG0 <- $RF[22\text{-}19]^{Ry}$ PC <- PC + 1	$XREG0_{IN}$ MA1=1, MA2=10, ALU=ADD, MPC=0, PC_{IN}
T2	R <- XREG0 $RF[22\text{-}19]^{Ry}$ <- XREG0 + exp	R_{IN}, MR1=1 cexp=0, ALU=ADD, MA1=0, MA2=01, MRF3=01, RFW3=1, MRF2=1
T3	DM[R] <- $RF[26\text{-}23]^{Rx}$	MRF1=1, DMW=1

INSTRUCTION	**DESCRIPTION**	**OPCODE**
STR R_x, [R_y + R_z]	**mem[R_y + R_z] = R_x**	**01001**
CLOCK	RTN	CONTROL SIGNALS
T0	IR <- IM [PC]	IR_{IN}, MOP=1
T1	XREG0 <- $RF[22\text{-}19]^{Ry}$ XREG1 <- $RF[18\text{-}15]^{Rz}$ PC <- PC + 1	$XREG0_{IN}$ $XREG1_{IN}$, MRF1=0 MA1=1, MA2=10, ALU=ADD, MPC=0, PC_{IN}
T2	R <- XREG0 + XREG1	R_{IN}, ALU=ADD, MA1=0, MA2=00, MR=0
T3	DM[R] <- $RF[26\text{-}23]^{Rx}$	MRF1=1, DMW=1

INSTRUCTION	**DESCRIPTION**	**OPCODE**
STR R_x, [R_y += R_z]	**mem[R_y + R_z] = R_x, R_y = R_y + R_z**	**01010**
CLOCK	RTN	CONTROL SIGNALS
T0	IR <- IM [PC]	IR_{IN}, MOP=1
T1	XREG0 <- $RF[22\text{-}19]^{Ry}$ XREG1 <- $RF[18\text{-}15]^{Rz}$ PC <- PC + 1	$XREG0_{IN}$ $XREG1_{IN}$, MRF1=0 MA1=1, MA2=10, ALU=ADD, MPC=0, PC_{IN}
T2	R <- XREG0 + XREG1 $RF[22\text{-}19]^{Ry}$ <- XREG0 + XREG1	R_{IN}, ALU=ADD, MA1=0, MA2=00, MR=0 MRF3=01, RFW3=1, MRF2=1
T3	DM[R] <- $RF[26\text{-}23]^{Rx}$	MRF1=1, DMW=1

INSTRUCTION	**DESCRIPTION**	**OPCODE**
STR R_x, [R_y], R_z	**mem[R_y] = R_x, R_y = R_y + R_z**	**01011**
CLOCK	RTN	CONTROL SIGNALS
T0	IR <- IM [PC]	IR_{IN}, MOP=1
T1	XREG0 <- $RF[22\text{-}19]^{Ry}$ XREG1 <- $RF[18\text{-}15]^{Rz}$ PC <- PC + 1	$XREG0_{IN}$ $XREG1_{IN}$, MRF1=0 MA1=1, MA2=10, ALU=ADD, MPC=0, PC_{IN}
T2	R <- XREG0 $RF[22\text{-}19]^{Ry}$ <- XREG0 + XREG1	R_{IN}, MR1=1 ALU=ADD, MA1=0, MA2=00, MRF3=01, RFW3=1, MRF2=1
T3	DM[R] <- $RF[26\text{-}23]^{Rx}$	MRF1=1, DMW=1

INSTRUCTION	DESCRIPTION	OPCODE
ADD/SUB R_x, R_y, #const	**$R_x = R_y$ +/- const**	**01100/01101**

CLOCK	RTN	CONTROL SIGNALS
T0	IR <- IM [PC]	IR_{IN}, MOP=1
T1	XREG0 <- RF[22-19]^{Ry} PC <- PC + 1	$XREG0_{IN}$ MA1=1, MA2=10, ALU=ADD, MPC=0, PC_{IN}
T2	R <- XREG0 +/- exp	R_{IN}, cexp=0, ALU=ADD/SUB, MA1=0, MA2=01, MR=0, SR_{IN}
T3	RF[26-23]^{Rx} <- R	MRF3=10, RFW3=1, MRF2=0

INSTRUCTION	DESCRIPTION	OPCODE
ADD/SUB R_x, R_y, R_z	**$R_x = R_y$ +/- R_z**	**01110/01111**

CLOCK	RTN	CONTROL SIGNALS
T0	IR <- IM [PC]	IR_{IN}, MOP=1
T1	XREG0 <- RF[22-19]^{Ry} XREG1 <- RF[18-15]^{Rz} PC <- PC + 1	$XREG0_{IN}$ $XREG1_{IN}$, MRF1=0 MA1=1, MA2=10, ALU=ADD, MPC=0, PC_{IN}
T2	R <- XREG0 +/- XREG1	R_{IN}, ALU=ADD/SUB, MA1=0, MA2=00, MR=0, SR_{IN}
T3	RF[26-23]^{Rx} <- R	MRF3=10, RFW3=1, MRF2=0

INSTRUCTION	DESCRIPTION	OPCODE
NOT/SHR/SHL R_x	**R_x = NOT/SHR/SHL R_x**	**10000/10001/10010**

CLOCK	RTN	CONTROL SIGNALS
T0	IR <- IM [PC]	IR_{IN}, MOP=1
T1	XREG1 <- RF[22-19]^{Ry} PC <- PC + 1	$XREG1_{IN}$, MRF1=1 MA1=1, MA2=10, ALU=ADD, MPC=0, PC_{IN}
T2	R <- NOT/SHR/SHL XREG1	R_{IN}, ALU=NOT/SHR/SHL, MA1=0, MA2=00, MR=0, SR_{IN}
T3	RF[26-23]^{Rx} <- R	MRF3=10, RFW3=1, MRF2=0

INSTRUCTION	DESCRIPTION	OPCODE
OR/AND/XOR R_x, R_y, #const	**$R_x = R_y$ OR/AND/XOR const**	**10011/10100/10101**

CLOCK	RTN	CONTROL SIGNALS
T0	IR <- IM [PC]	IR_{IN}, MOP=1
T1	XREG0 <- RF[22-19]^{Ry} PC <- PC + 1	$XREG0_{IN}$ MA1=1, MA2=10, ALU=ADD, MPC=0, PC_{IN}
T2	R <- XREG0 OR/AND/XOR exp	R_{IN}, cexp=0, ALU=OR/AND/XOR, MA1=0, MA2=01, MR=0, SR_{IN}
T3	RF[26-23]^{Rx} <- R	MRF3=10, RFW3=1, MRF2=0

INSTRUCTION	DESCRIPTION	OPCODE
OR/AND/XOR R_x, R_y, R_z	**R_x = R_y OR/AND/XOR R_z**	**10110/10111/11000**
CLOCK	RTN	CONTROL SIGNALS
T0	IR <- IM [PC]	IR_{IN}, MOP=1
T1	XREG0 <- RF[22-19]^{Ry} XREG1 <- RF[18-15]^{Rz} PC <- PC + 1	$XREG0_{IN}$ $XREG1_{IN}$, MRF1=0 MA1=1, MA2=10, ALU=ADD, MPC=0, PC_{IN}
T2	R <- XREG0 OR/AND/XOR XREG1	R_{IN}, ALU= OR/AND/XOR, MA1=0, MA2=00, MR=0, SR_{IN}
T3	RF[26-23]^{Rx} <- R	MRF3=10, RFW3=1, MRF2=0

INSTRUCTION	DESCRIPTION	OPCODE
MOV R_x, R_y	**R_x = R_y**	**11001**
CLOCK	RTN	CONTROL SIGNALS
T0	IR <- IM [PC]	IR_{IN}, MOP=1
T1	XREG0 <- RF[22-19]^{Ry} PC <- PC + 1	$XREG0_{IN}$ MA1=1, MA2=10, ALU=ADD, MPC=0, PC_{IN}
T2	RF[26-23]^{Rx} <- XREG0	MR=1MRF3=11, RFW3=1, MRF2=0

INSTRUCTION	DESCRIPTION	OPCODE
JMP label	**PC = adr(label)**	**11010**
CLOCK	RTN	CONTROL SIGNALS
T0	IR <- IM [PC]	IR_{IN}, MOP=1
T1	PC <- exp	PC_{IN}, MPC=1, cexp=1

INSTRUCTION	DESCRIPTION	OPCODE
JZ label	**if(SR<0> = 1) -> PC = adr(label)**	**11011**
CLOCK	RTN	CONTROL SIGNALS
T0	IR <- IM [PC]	IR_{IN}, MOP=1
T1	SR<0>=1 -> PC <- exp	PC_{IN}, MPC=1, cexp=1

INSTRUCTION	DESCRIPTION	OPCODE
JC label	**if(SR<1> = 1) -> PC = adr(label)**	**11100**
CLOCK	RTN	CONTROL SIGNALS
T0	IR <- IM [PC]	IR_{IN}, MOP=1
T1	SR<1>=1 -> PC <- exp	PC_{IN}, MPC=1, cexp=1

INSTRUCTION	DESCRIPTION	OPCODE
NOP	**no operation**	**11101**
CLOCK	RTN	CONTROL SIGNALS
T0	IR <- IM [PC]	IR_{IN}, MOP=1
T1	PC <- PC + 1	MA1=1, MA2=10, ALU=ADD, MPC=0, PC_{IN}

INSTRUCTION		DESCRIPTION		OPCODE
END		**end of program**		**11110**
CLOCK	RTN		CONTROL SIGNALS	
T0	IR <- IM [PC]		IR_{IN}, MOP=1	

ASSEMBLER INSTRUCTION		EQUIVALENT INSTRUCTION
LDR R_x, var	⇔	**LDR R_x, [R_{15}, @var]**
LDR R_x, #const	⇔	**LDR R_x, [R_{15}, @const]**
STR var, R_x	⇔	**STR R_x, [R_{15}, @var]**
CLR R_x	⇔	**MOV R_x, R_{15}**

R_{15} is a 0 register, it contains the value 0x00000000

Control signals

Control vector:

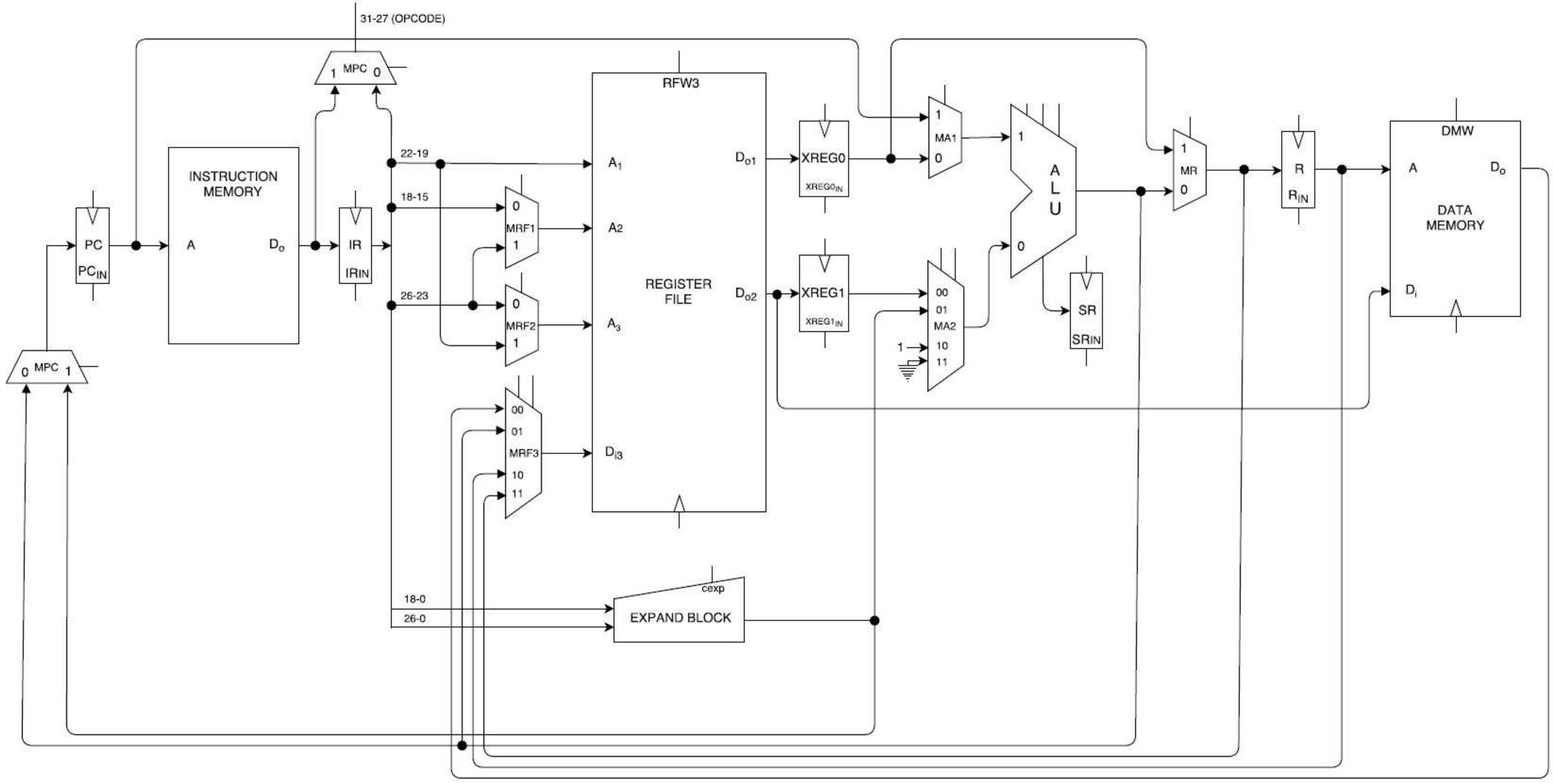
21-bit control vector

		MOP	SR_{IN}	PC_{IN}	MPC	IR_{IN}	MRF1
		MSB - 21	20	19	18	17	16
MRF2	$MRF3_1$	$MRF3_0$	RFW3	EXP	$XREG0_{IN}$	$XREG1_{IN}$	MA1
15	14	13	12	11	10	9	8
$MA2_1$	$MA2_0$	ALU_2	ALU_1	ALU_0	MR	R_{IN}	DMW
7	6	5	4	3	2	1	LSB - 0

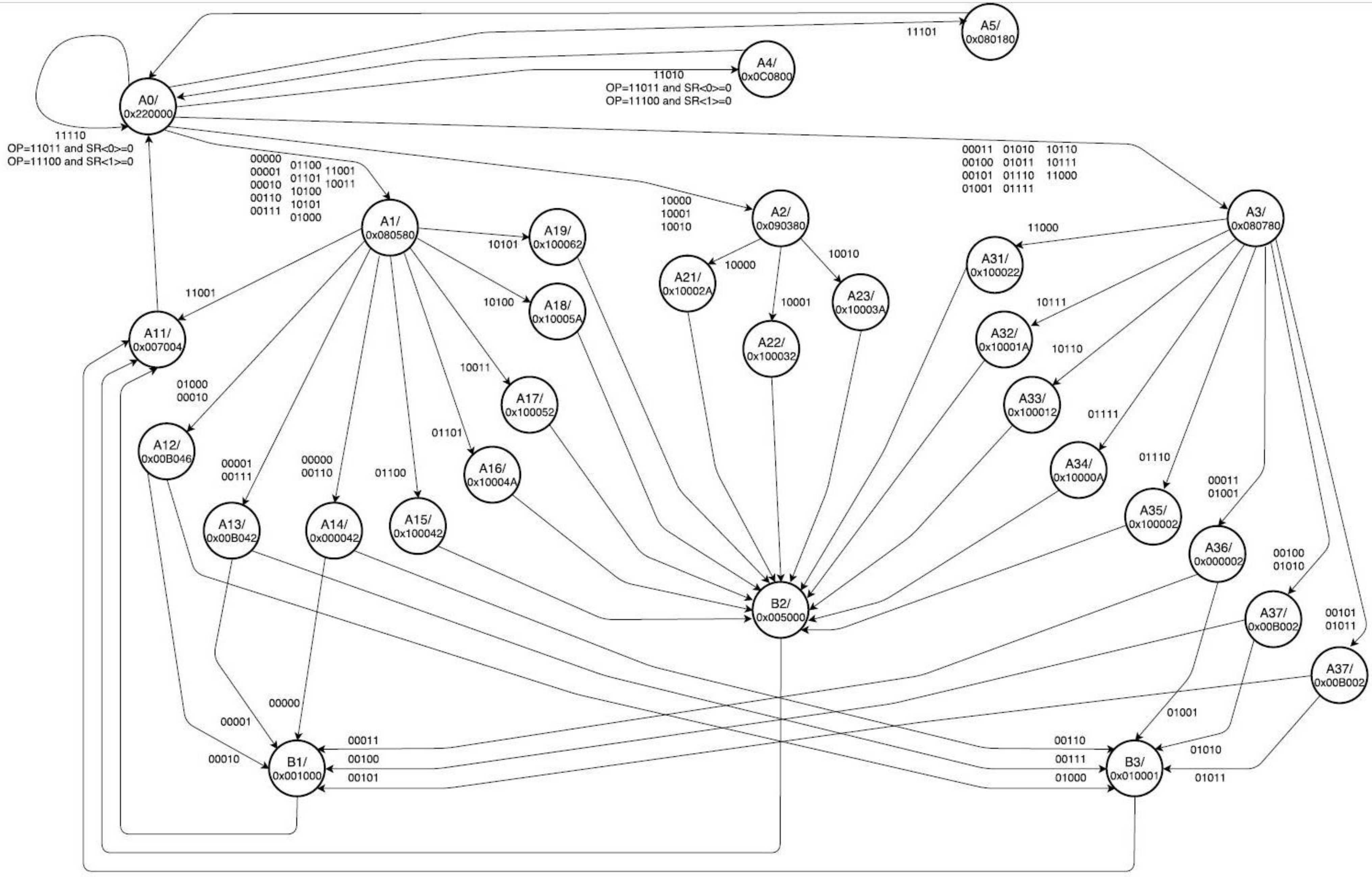
ALU functions

ALU_2	ALU_1	ALU_0	FUCTION
0	0	0	ADD
0	0	1	SUB
0	1	0	OR
0	1	1	AND
1	0	0	XOR
1	0	1	NOT(input 0)
1	1	0	SHR(input 0)
1	1	1	SHL(input 0)

DATAPATH



31-27 (OPCODE)
MPC
1
0
PC
PCIN
INSTRUCTION MEMORY
A
Do
IR
IRIN
22-19
18-15
26-23
18-0
26-0
MRF1
MRF2
MRF3
00
01
10
11
RFW3
A1
A2
A3
Di3
REGISTER FILE
Do1
Do2
EXPAND BLOCK
cexp
XREG0
XREG0IN
XREG1
XREG1IN
MA1
MA2
ALU
SR
SRIN
MR
R
RIN
DMW
DATA MEMORY
Di



A0/
0x220000
11110
OP=11011 and SR<0>=0
OP=11100 and SR<1>=0
A4/
0x0C0800
11010
OP=11011 and SR<0>=0
OP=11100 and SR<1>=0
A5/
0x080180
11101
00000
00001
00010
00110
00111
01100
01101
10100
10101
01000
11001
10011
A1/
0x080580
10000
10001
10010
A2/
0x090380
00011 01010 10110
00100 01011 10111
00101 01110 11000
01001 01111
A3/
0x080780
A11/
0x007004
11001
A12/
0x00B046
01000
00010
A13/
0x00B042
00001
00111
A14/
0x000042
00000
00110
A15/
0x100042
01100
A16/
0x10004A
01101
A17/
0x100052
10011
A18/
0x10005A
10100
A19/
0x100062
10101
A21/
0x10002A
10000
A22/
0x100032
10001
A23/
0x10003A
10010
A31/
0x100022
11000
A32/
0x10001A
10111
A33/
0x100012
10110
A34/
0x10000A
01111
A35/
0x100002
01110
A36/
0x000002
00011
01001
A37/
0x00B002
00100
01010
A37/
0x00B002
00101
01011
B1/
0x001000
B2/
0x005000
B3/
0x010001
00000
00001
00010
00011
00100
00101
00110
00111
01000
01001
01010
01011