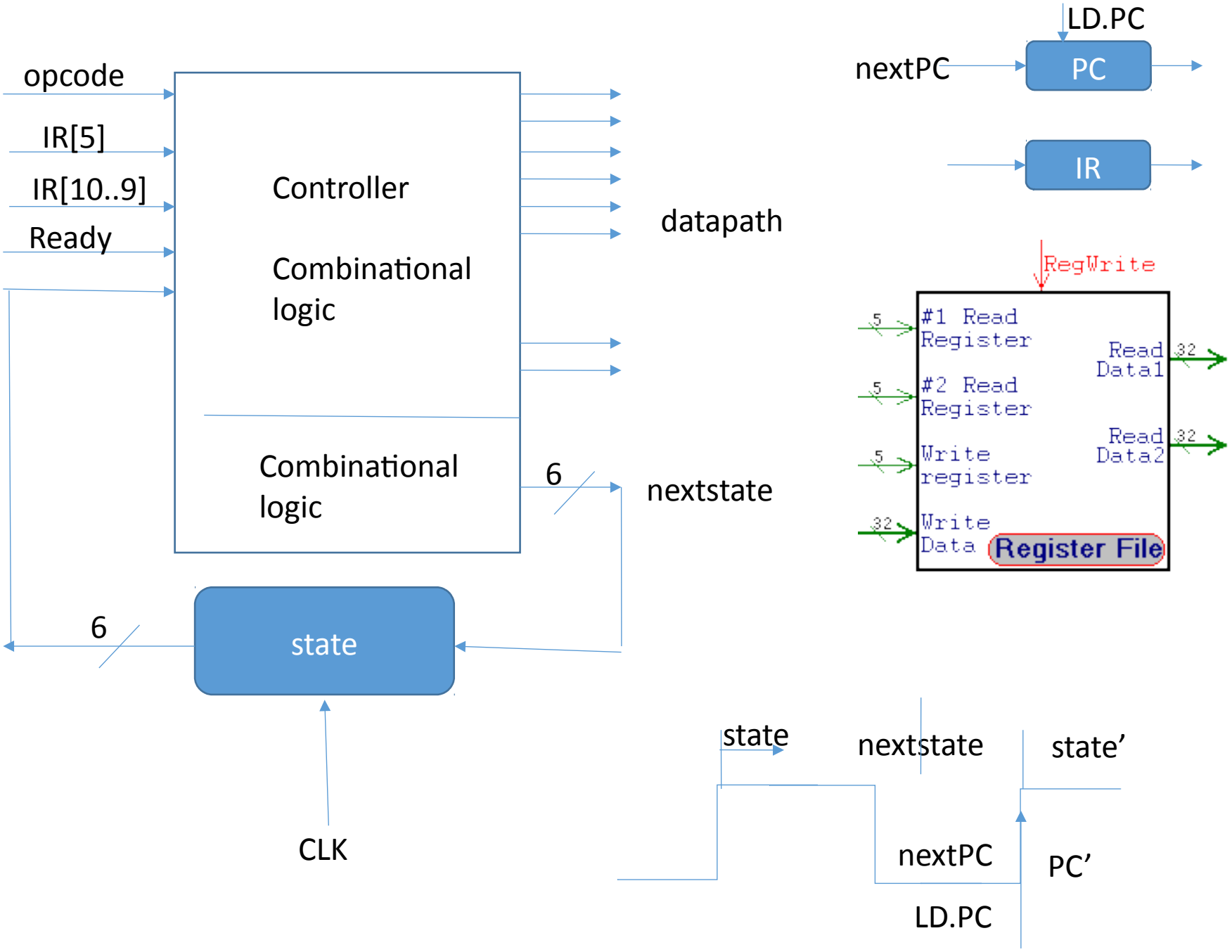
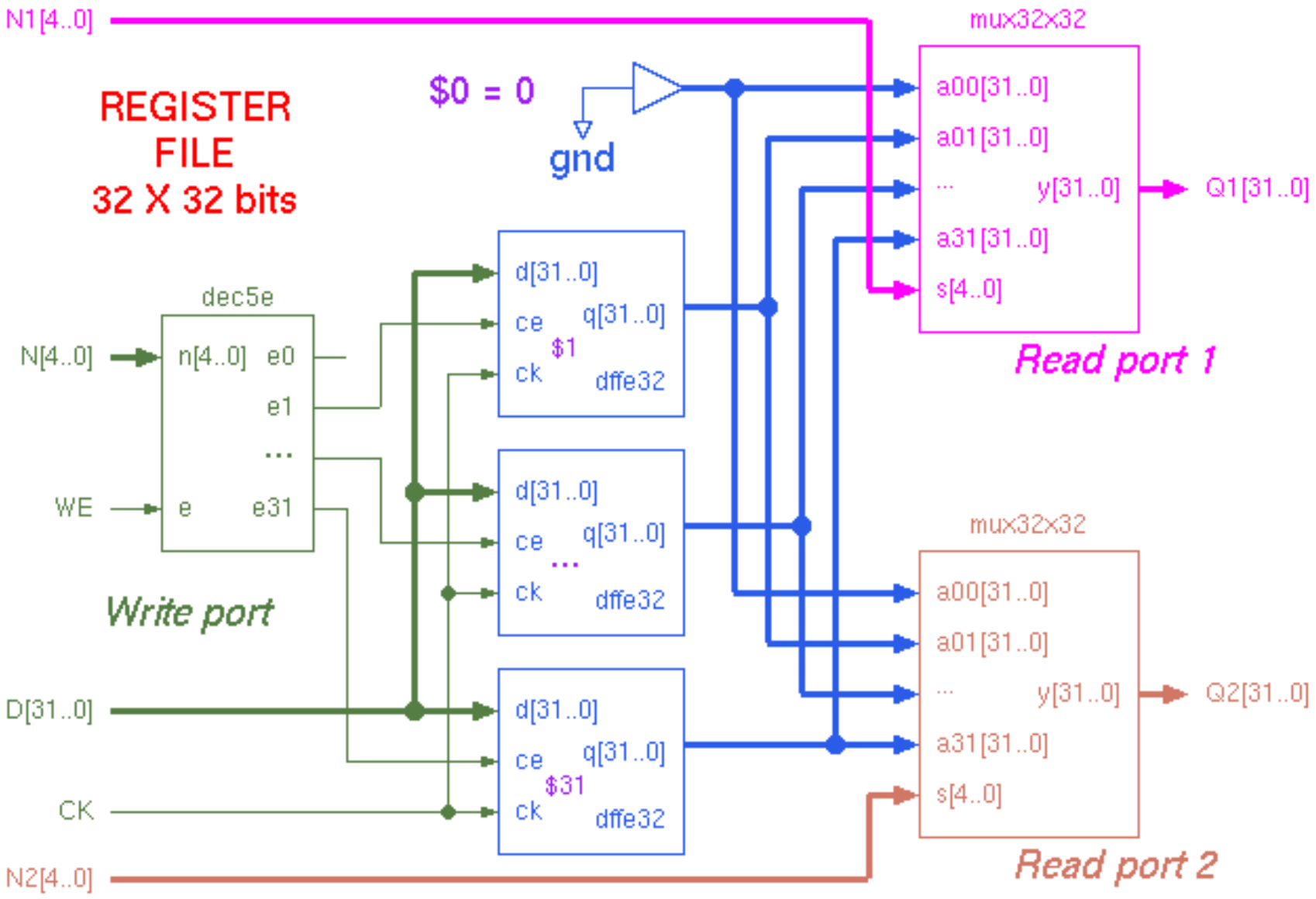
LC3 Instructions & Data Path &controller

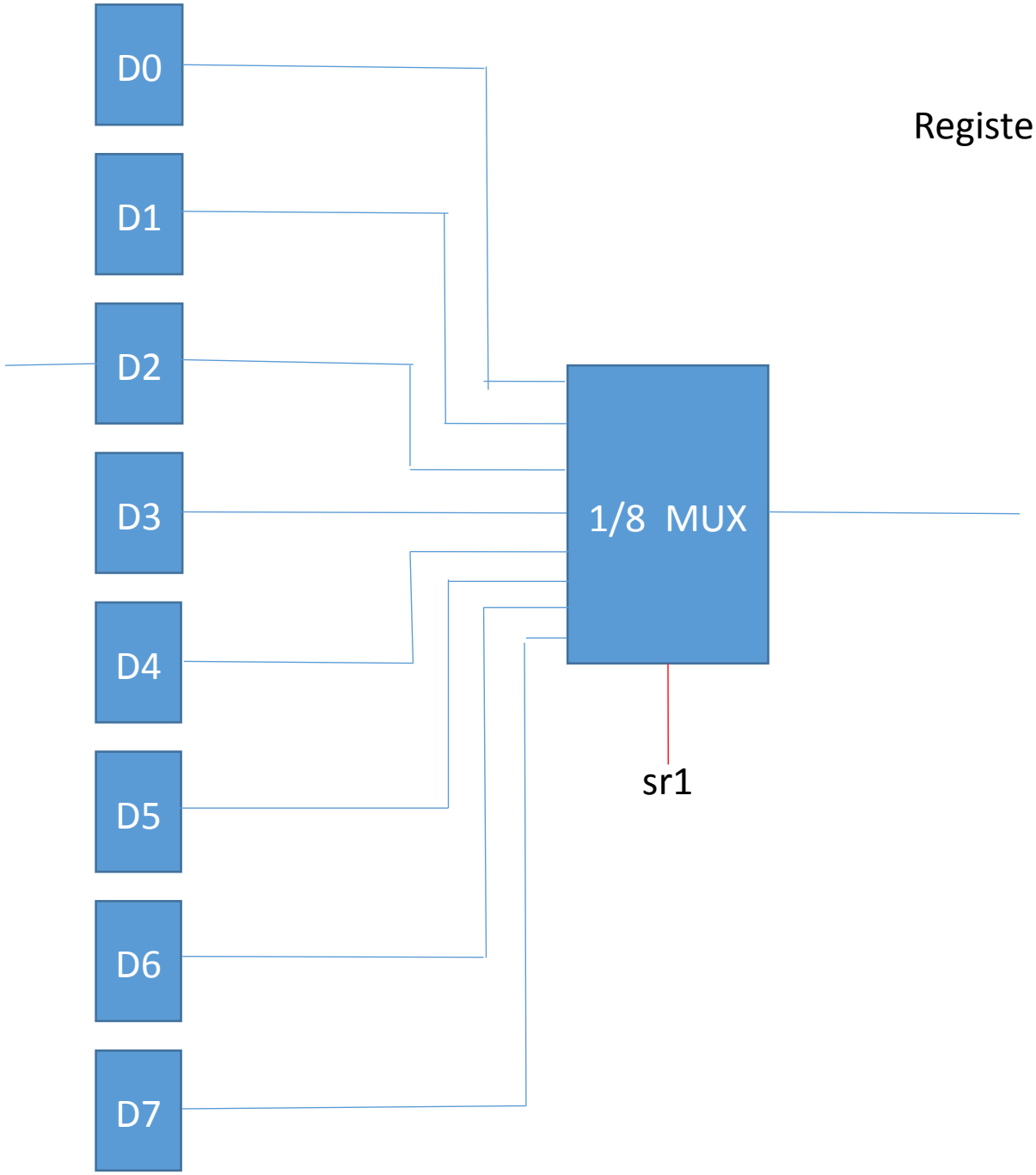
JIANG Xiaohong

jiangxh@zju.edu.cn

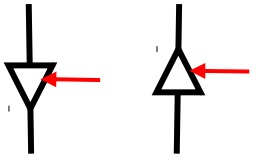

opcode
IR[5]
IR[10..9]
Ready
Controller
Combinational logic
Combinational logic
datapath
6
nextstate
6
state
CLK
LD.PC
nextPC
PC
IR
RegWrite
5
#1 Read Register
5
#2 Read Register
5
Write register
32
Write Data
Register File
Read Data1
32
Read Data2
32
state
nextstate
state'
nextPC
PC'
LD.PC

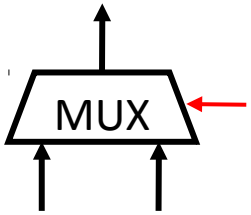
Register File

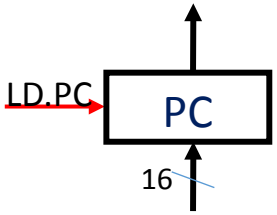


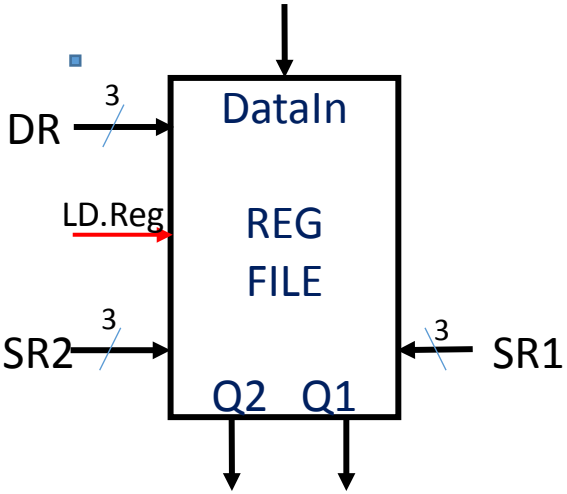


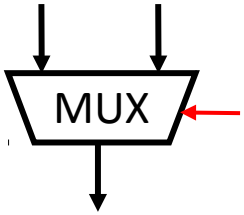
Register Read Port

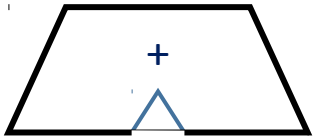



MUX

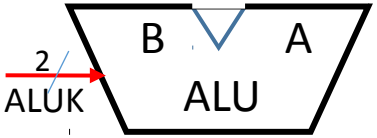

LD.PC
PC
16

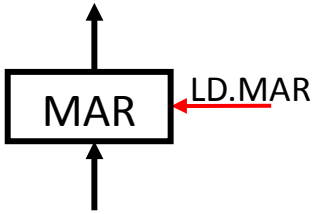

DR
3
DataIn
LD.Reg
REG
FILE
SR2
3
3
SR1
Q2
Q1

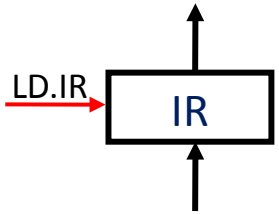

MUX

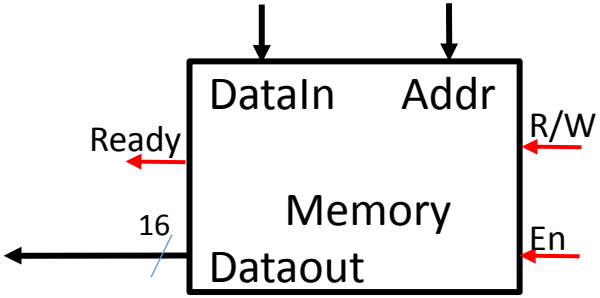

+

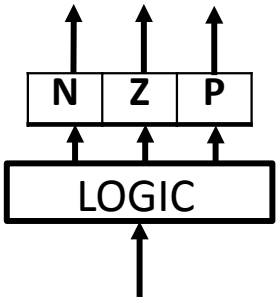

+1


B
A
2
ALUK
ALU

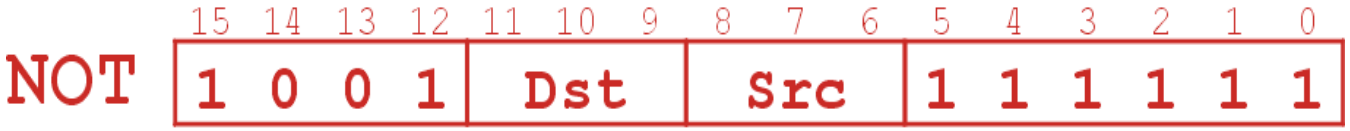

MAR
LD.MAR


LD.IR
IR

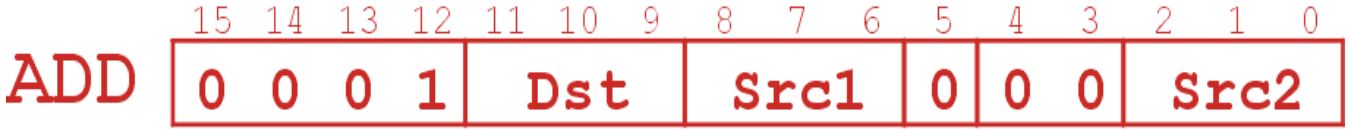

DataIn
Addr
R/W
Ready
Memory
16
En
Dataout


N
Z
P
LOGIC

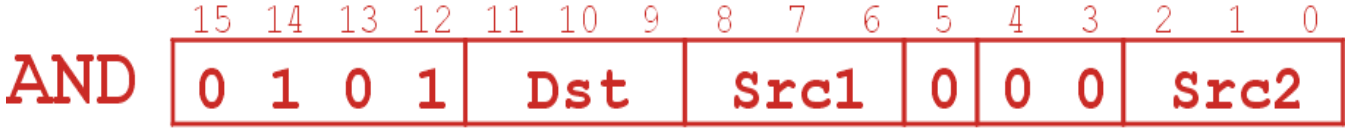
LC3 Instructions



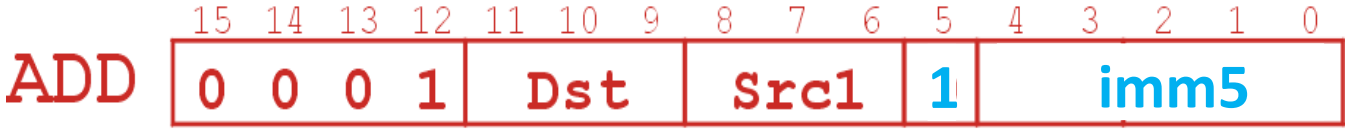
NOT R2, R1



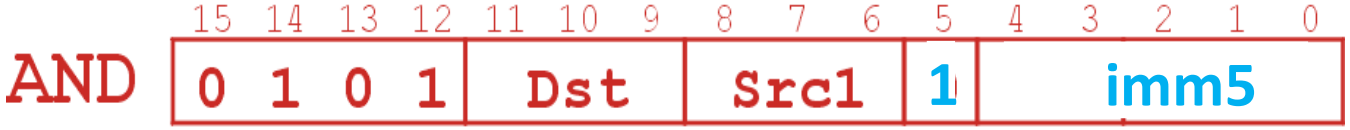
ADD R3, R1, R2



AND R3, R1, R2



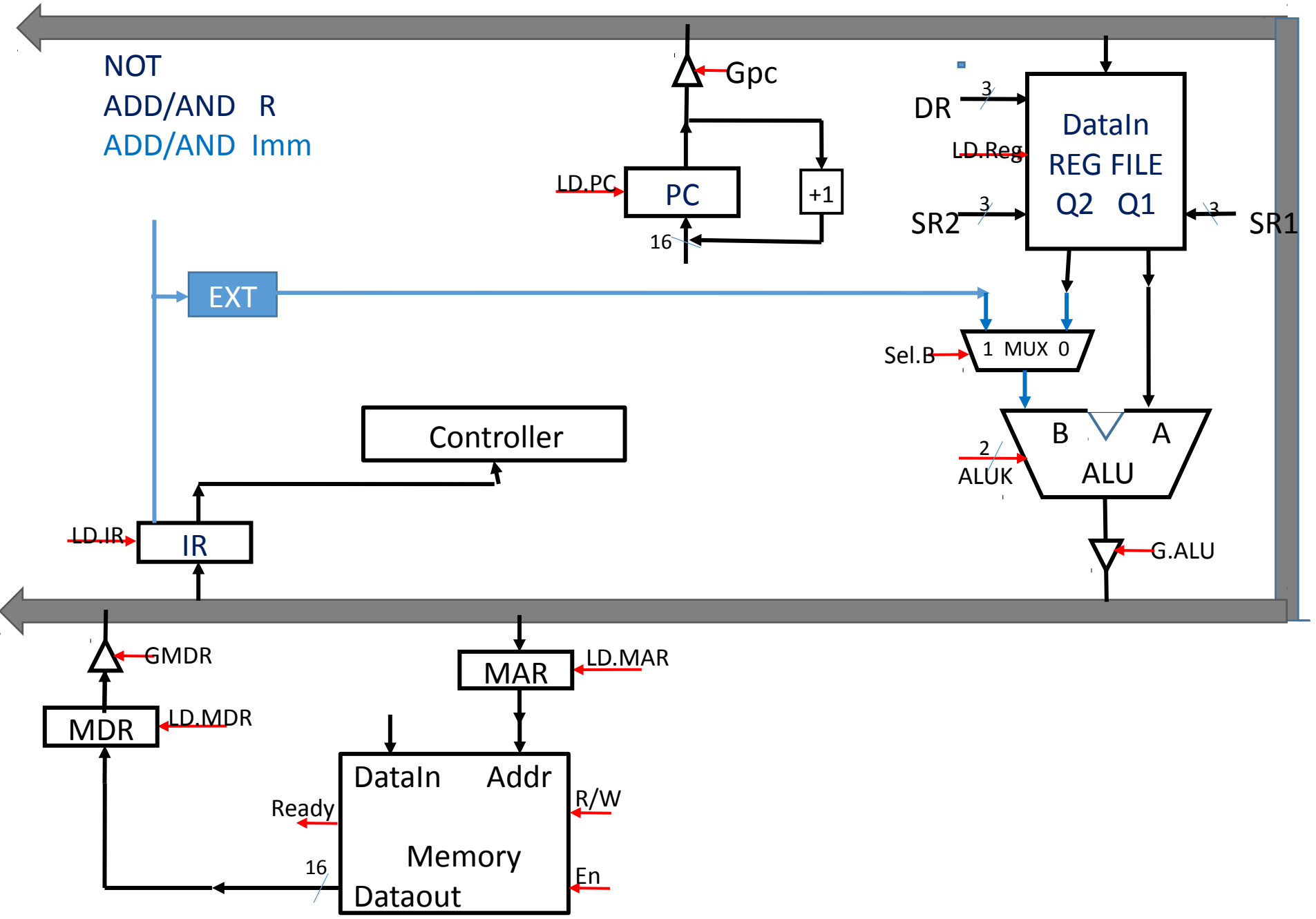
ADD R3, R1, #4



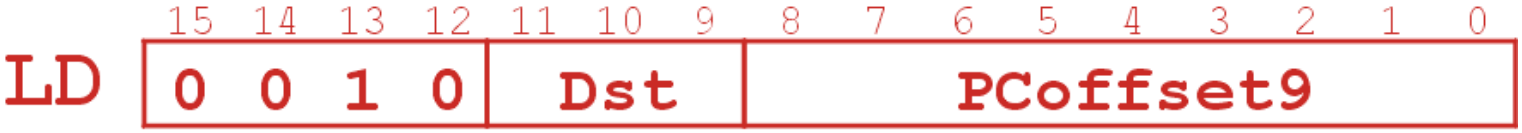
AND R3, R1, #4

Truth table for Control signals

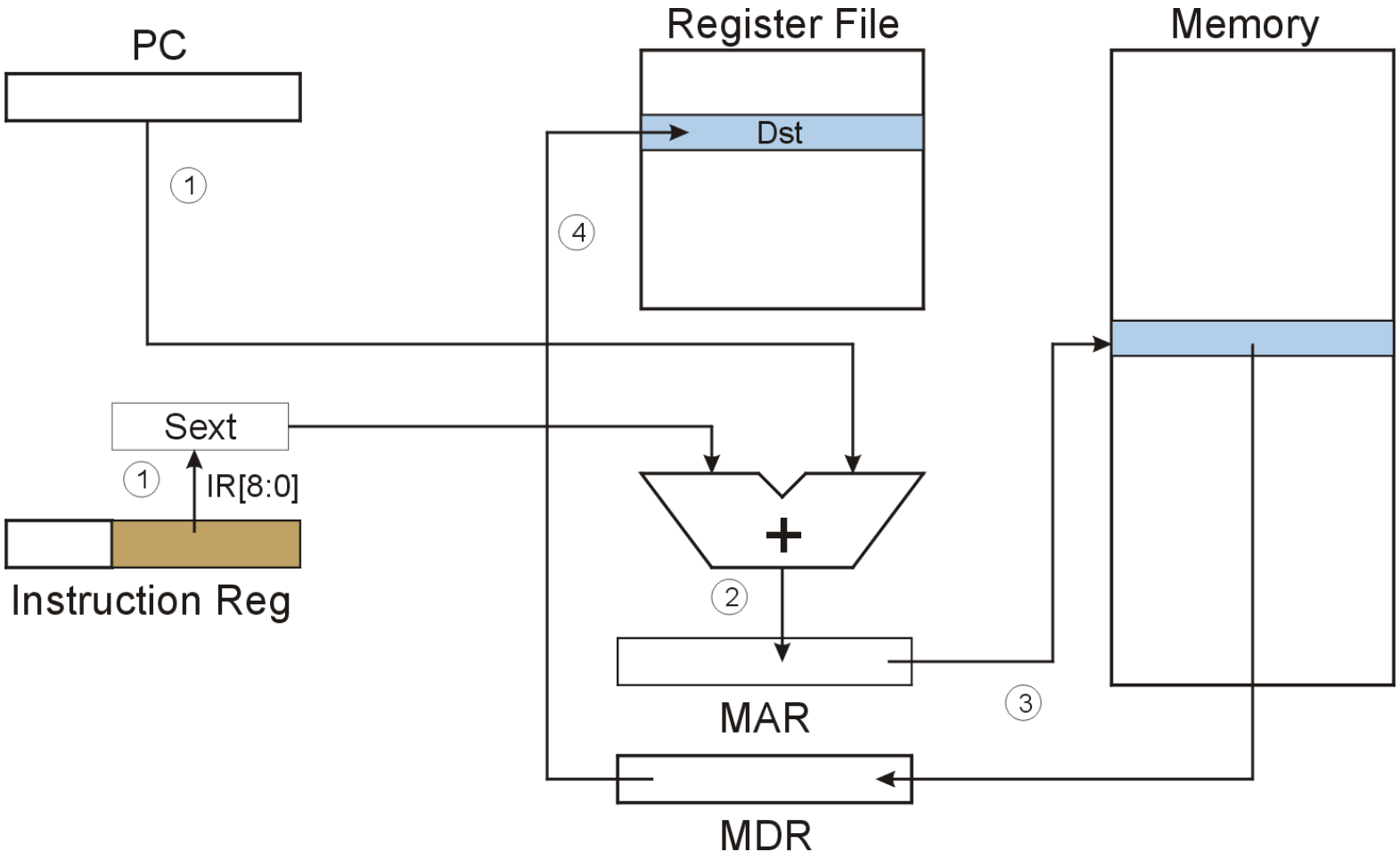
		Gpc	LD.MAR	LD.PC	R/W	En	LD.MDR	G.MDR	LD.IR	GALU	ALUK	LD.Reg	Sel.B	GmARMUX	SR1MUX	MDRMUx			
Instr. Fetch	S18	1	1	1											0	0			
	S33				0	1	1								0	0			
	S35							1	1						0	0			
Decode	S32														0	0			
NOT	S9									1	NOT	1			0	0			
+/*R	S1									1	+/*	1			0	0			
+/* i	S5									1	+/*	1	1		0	0			

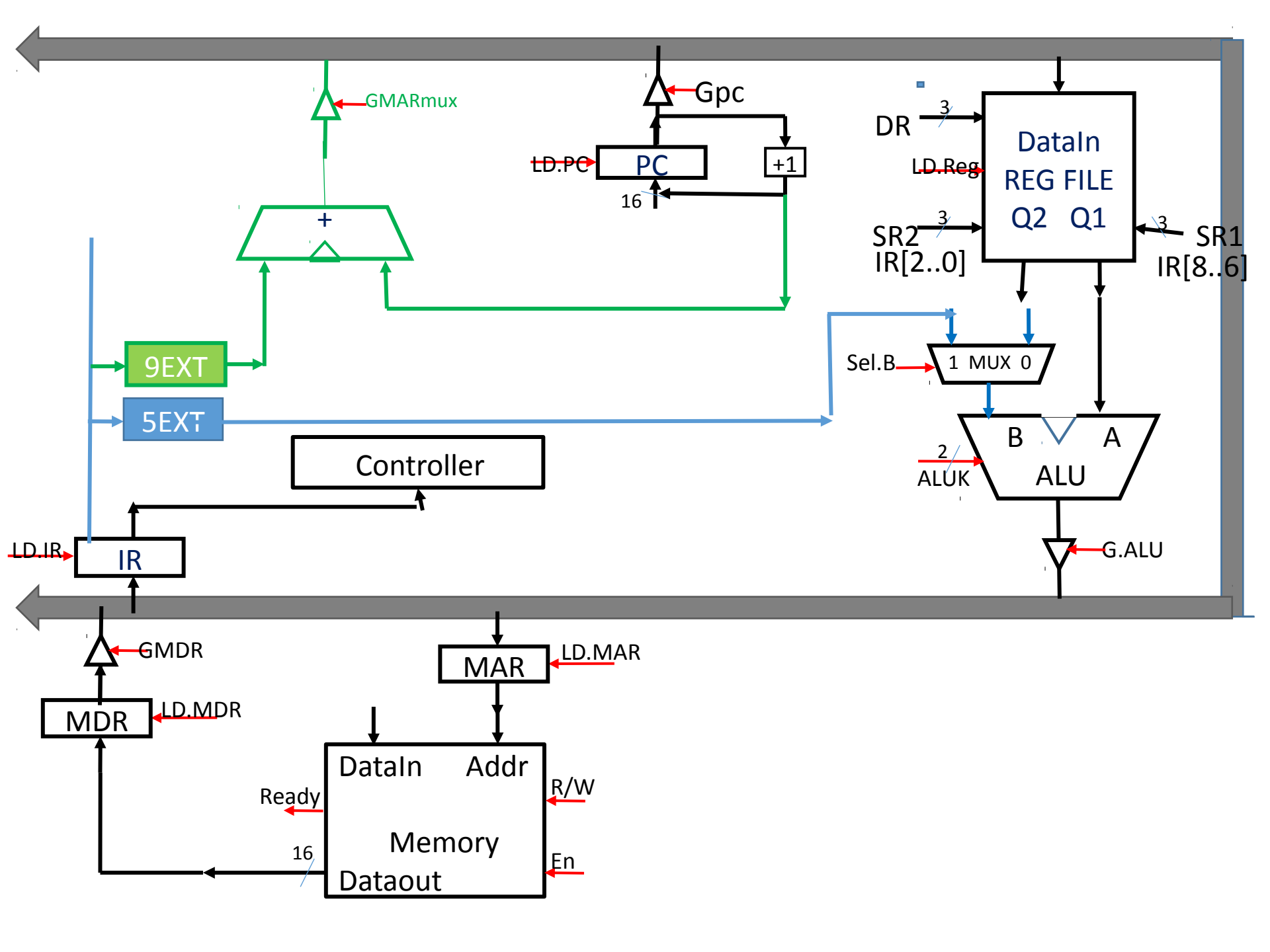

NOT
ADD/AND R
ADD/AND Imm
Gpc
LD.PC
PC
+1
16
DR
3
LD.Reg
DataIn
REG FILE
Q2 Q1
SR2
3
3
SR1
EXT
Sel.B
1 MUX 0
Controller
B
A
2
ALUK
ALU
LD.IR
IR
G.ALU
GMDR
MAR
LD.MAR
MDR
LD.MDR
DataIn
Addr
Ready
R/W
Memory
16
Dataout
En

LD (PC-Relative)

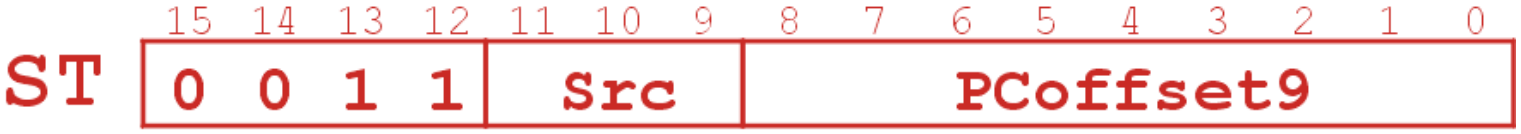


LD R1, PCoffset ; <- R1 Memory[PC + Pcoffset]

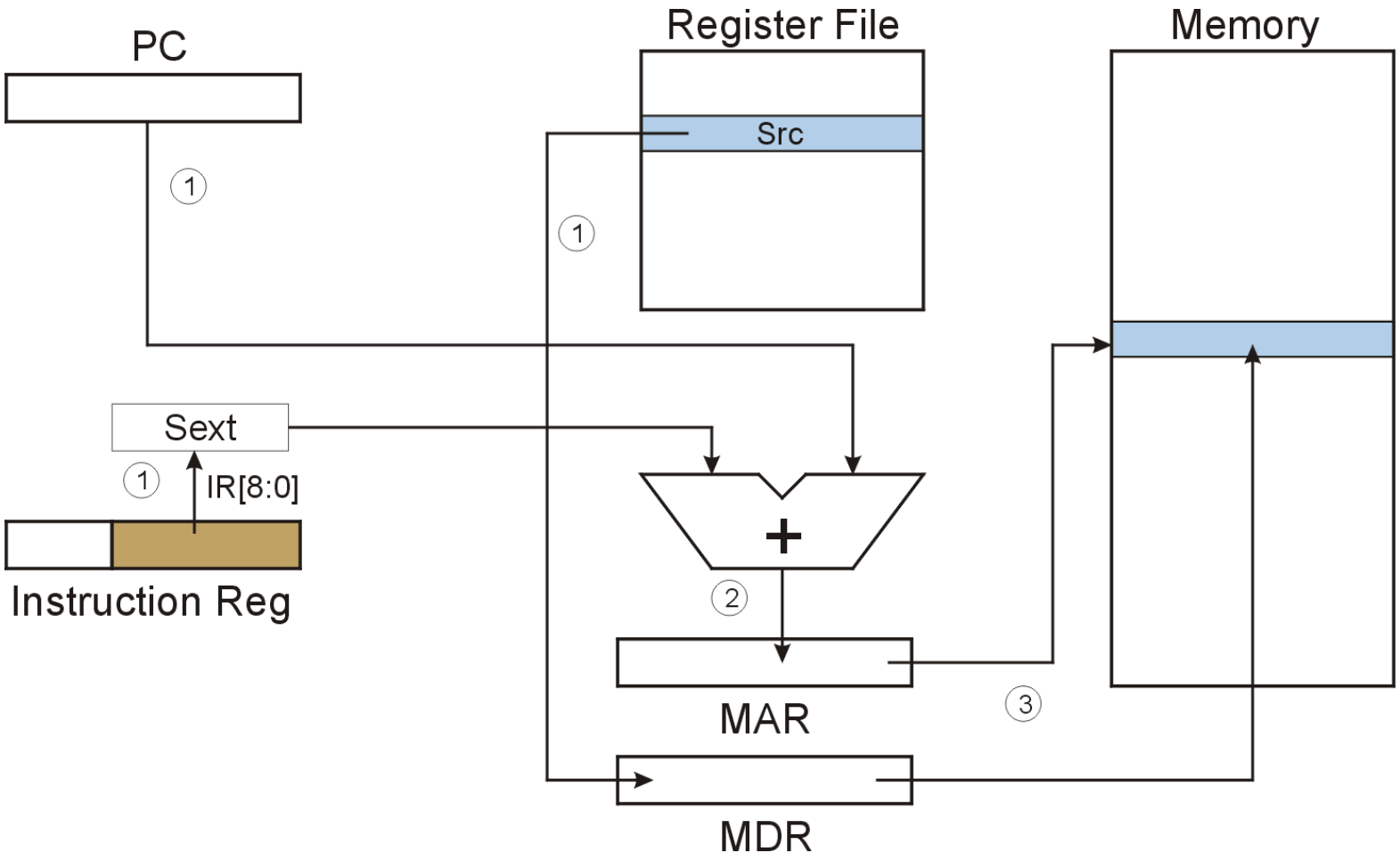



GMARmux
Gpc
LD.PC
PC
+1
16
+
DR
3
LD.Reg
DataIn
REG FILE
Q2 Q1
SR2
IR[2..0]
SR1
IR[8..6]
9EXT
5EXT
Sel.B
1 MUX 0
B
A
2
ALUK
ALU
G.ALU
Controller
LD.IR
IR
GMDR
MDR
LD.MDR
MAR
LD.MAR
DataIn
Addr
R/W
Ready
Memory
16
En
Dataout

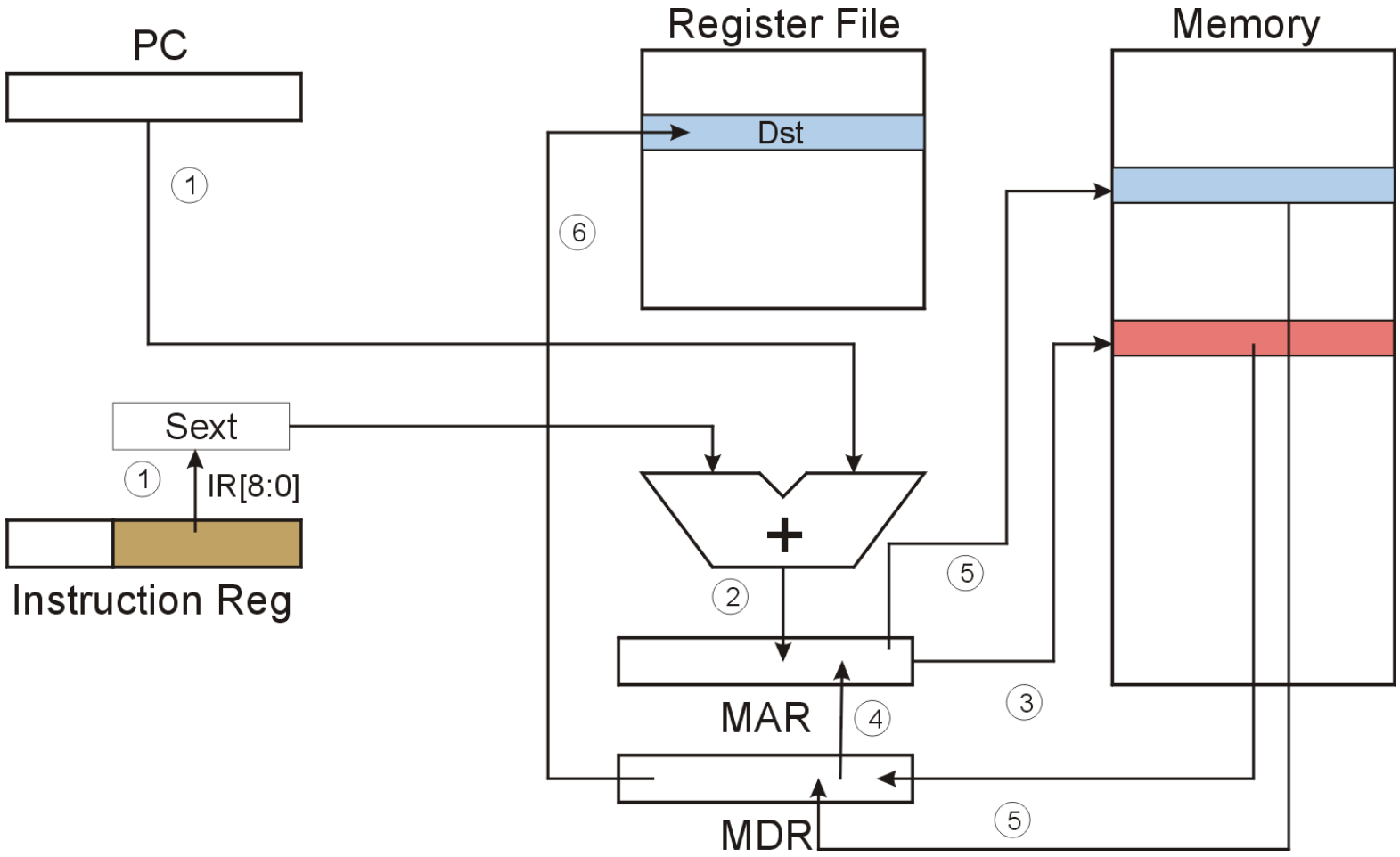
ST (PC-Relative)



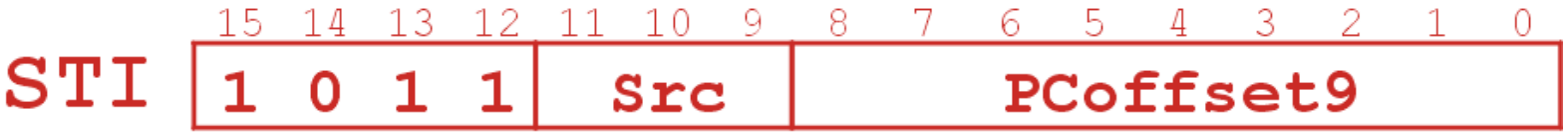
ST R1, PCoffset ;Memory[PC + Pcoffset] <- R1

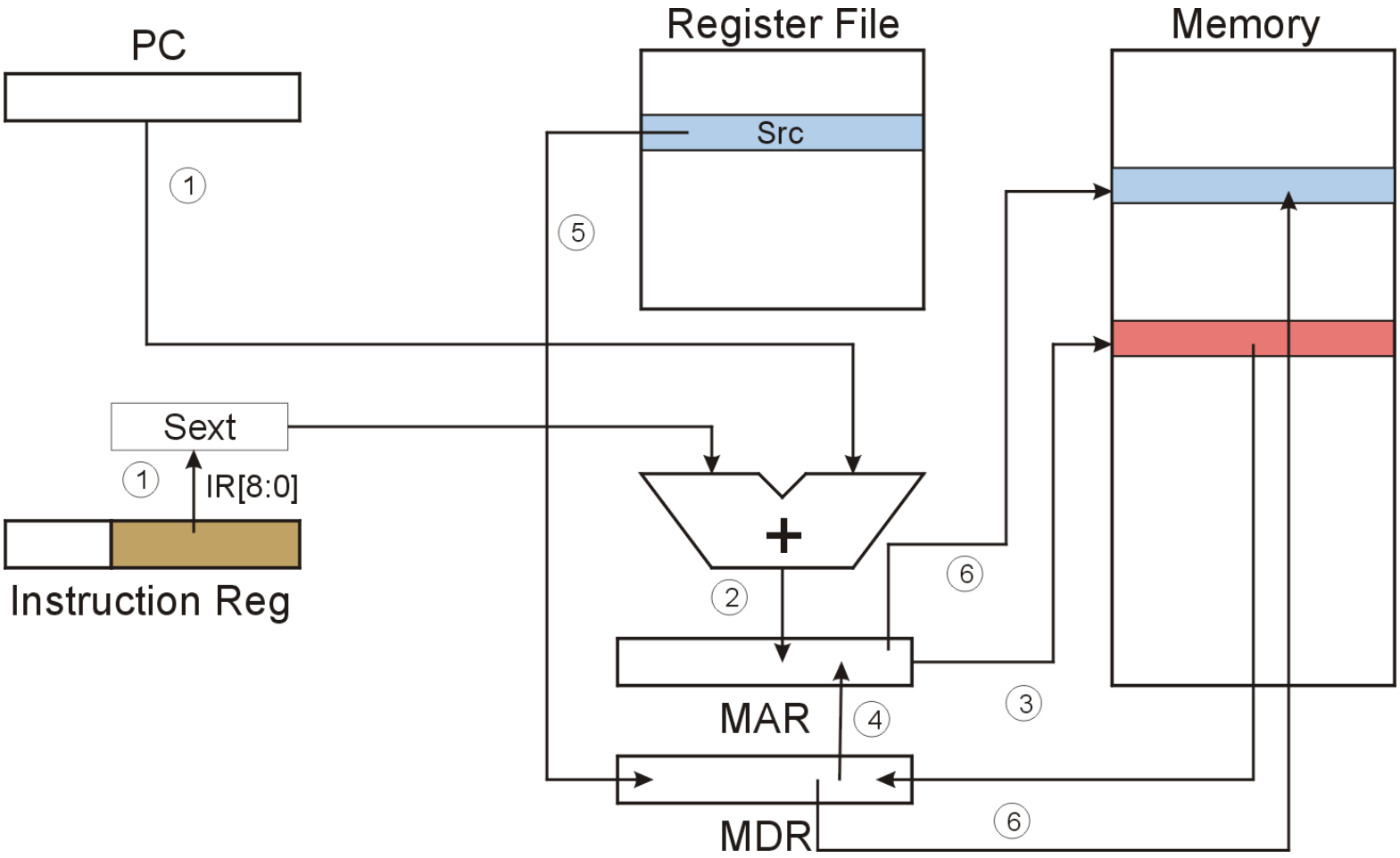


LDI (Indirect)



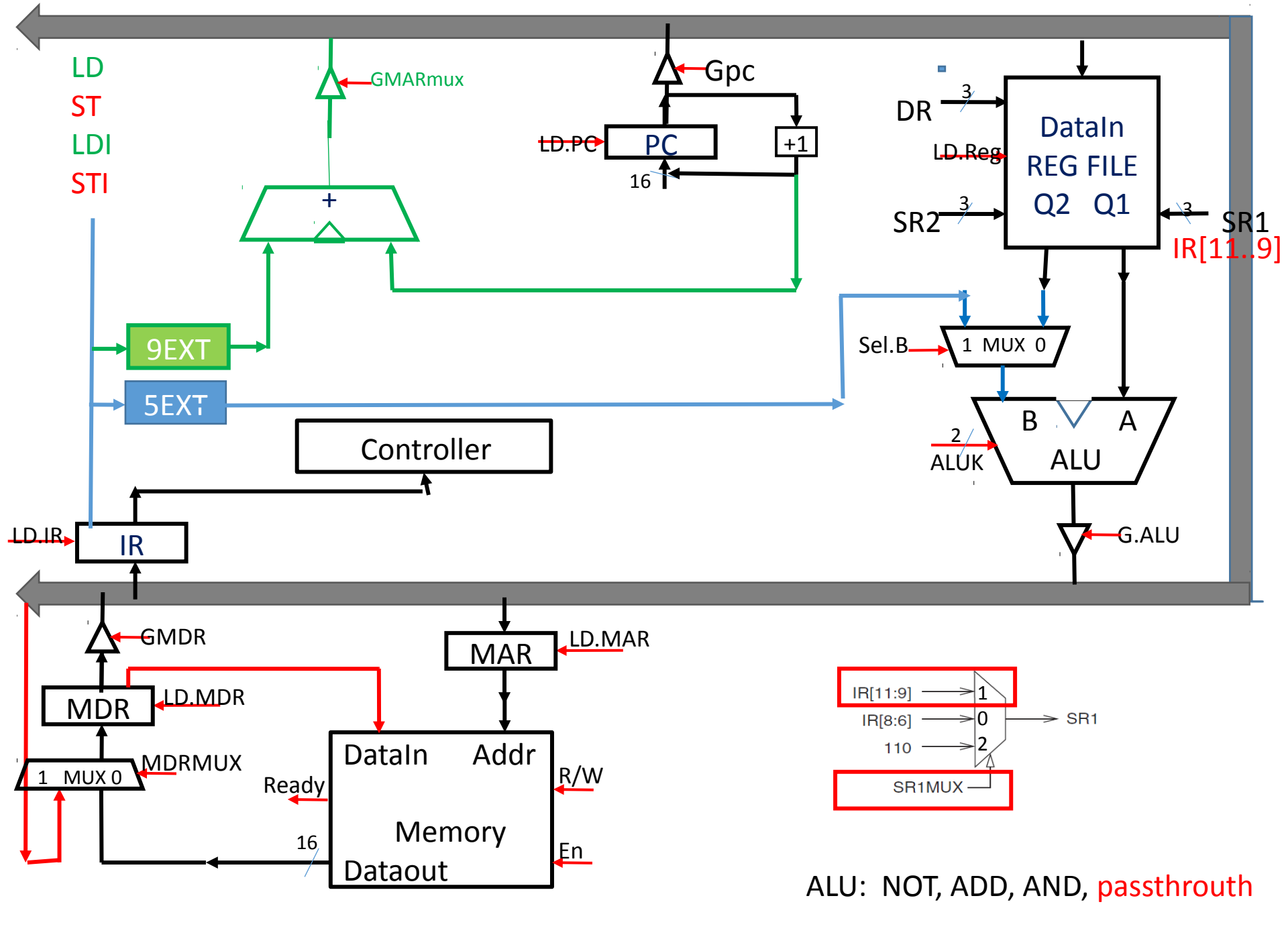
STI (Indirect)





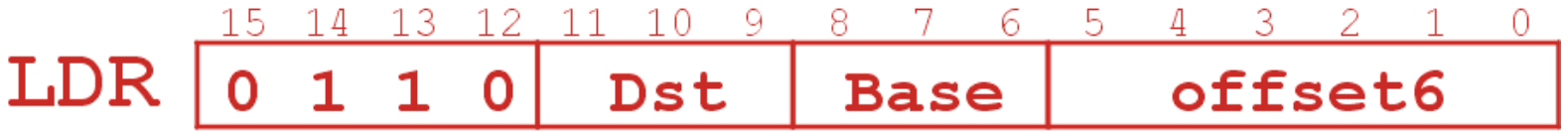
Truth table for Control signals

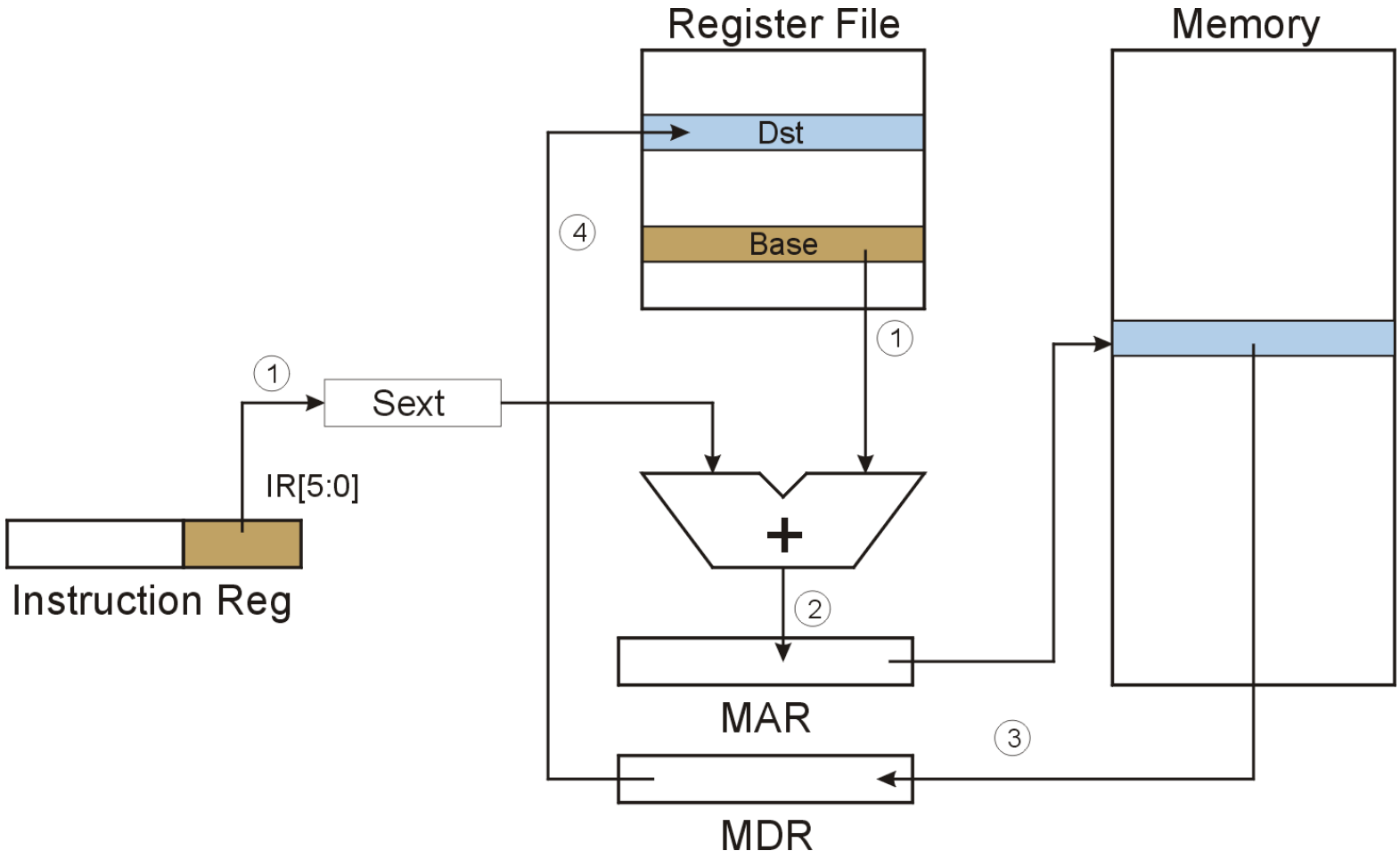
		Gpc	LD.MAR	LD.PC	R/W	En	LD.MDR	G.MDR	LD.IR	GALU	ALUK	LD.Reg	Sel.B	GmARMUX	SR1MUX	MDRMUx			
Instr. Fetch	S18	1	1	1											0	0			
	S33				0	1	1								0	0			
	S35							1	1						0	0			
Decode	S32														0	0			
NOT	S9									1	NOT	1			0	0			
+/*R	S1									1	+/*	1			0	0			
+/* i	S5									1	+/*	1	1		0	0			
LD/ LDI/ ST/STI	S2/S3/ S10/s11		1											1	0	0			
LD /LDR /LDI	S25				0	1	1								0	0			
	S27							1				1			0	0			
ST /STR /STI	S23							1		1					1	1			
	S16				1	1													
LDI/STI	S26/S31		1					1											



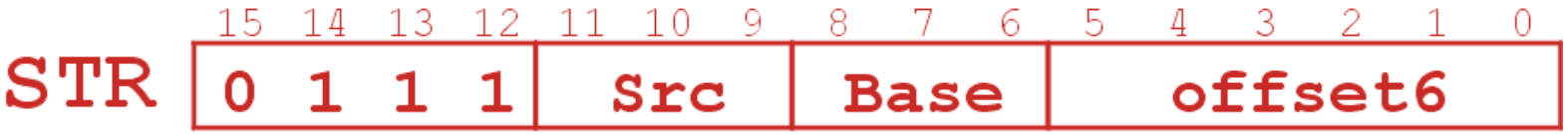
ALU: NOT, ADD, AND, passthrouth

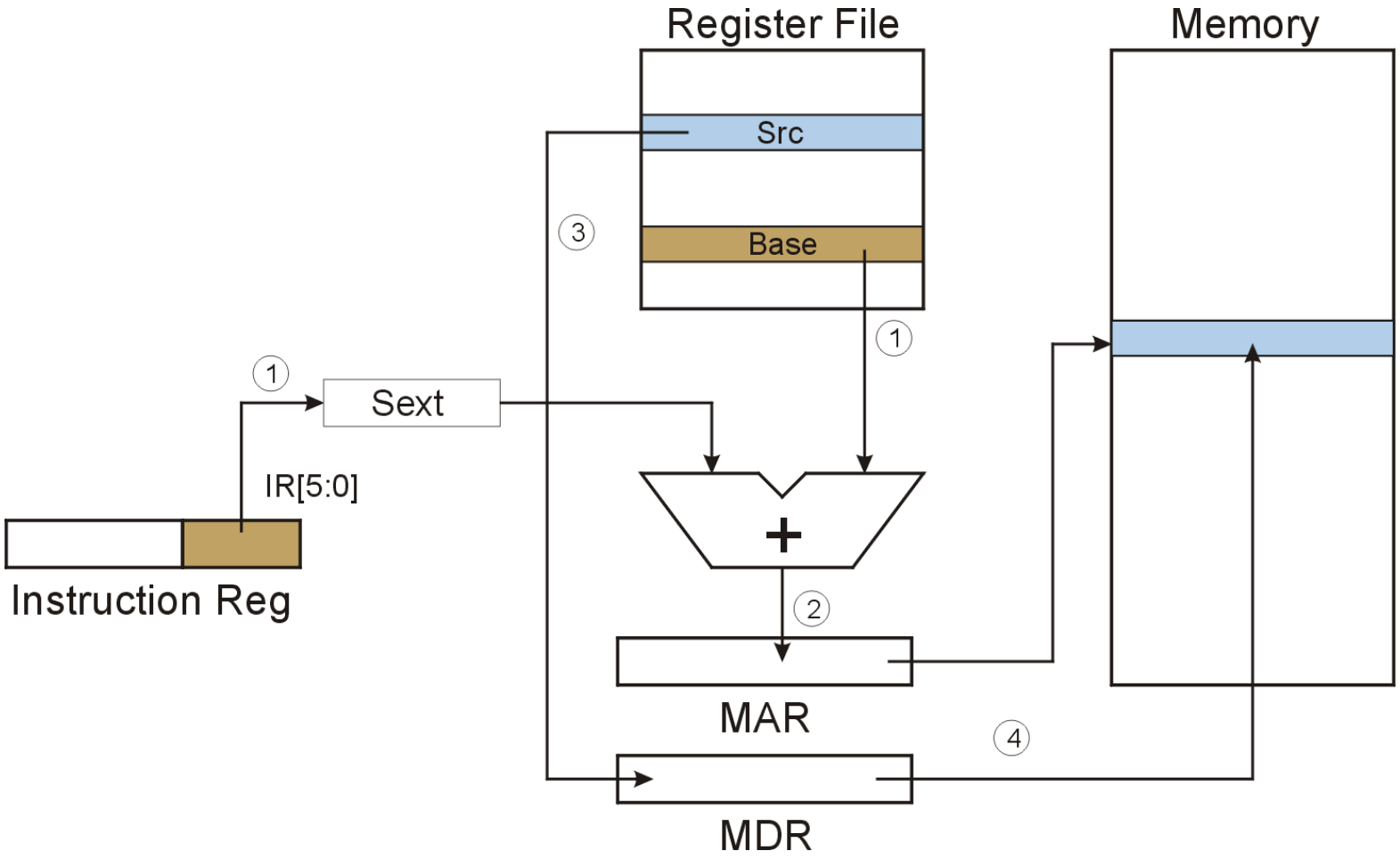
LDR (Base+Offset)



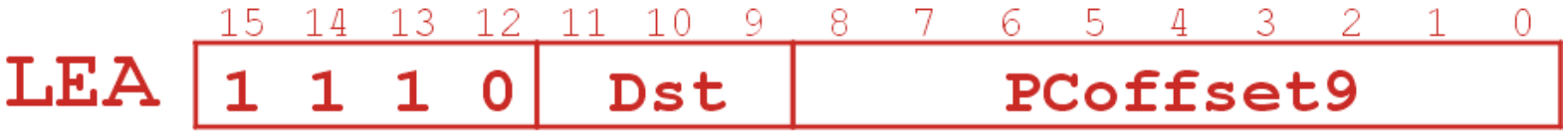


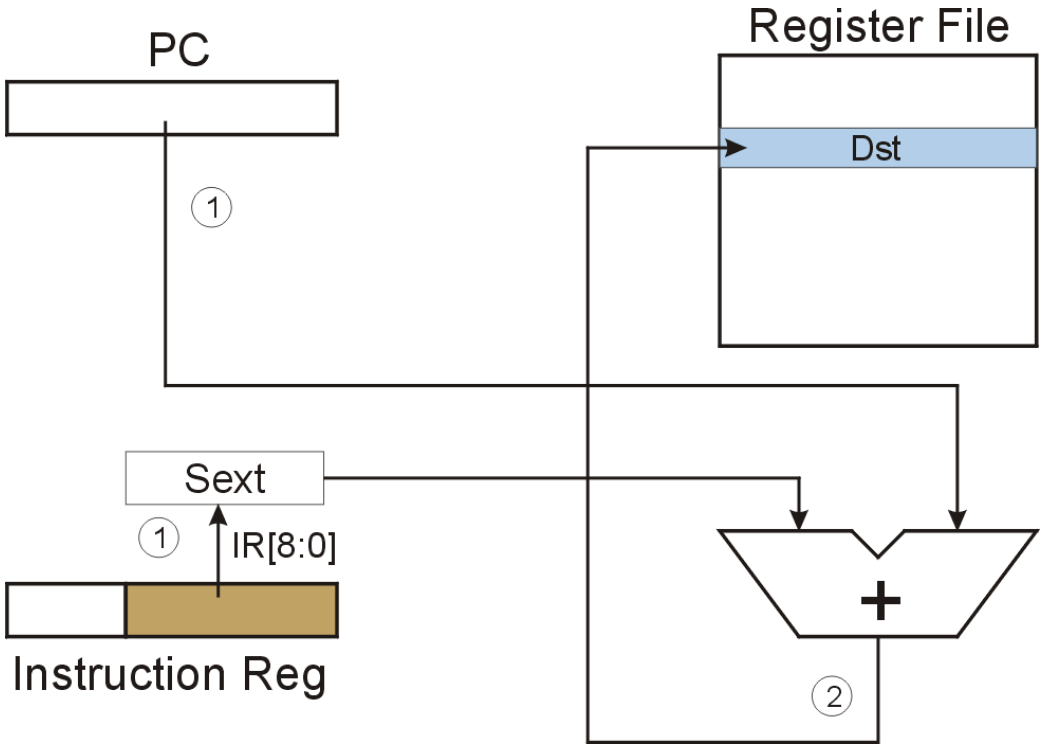
STR (Base+Offset)





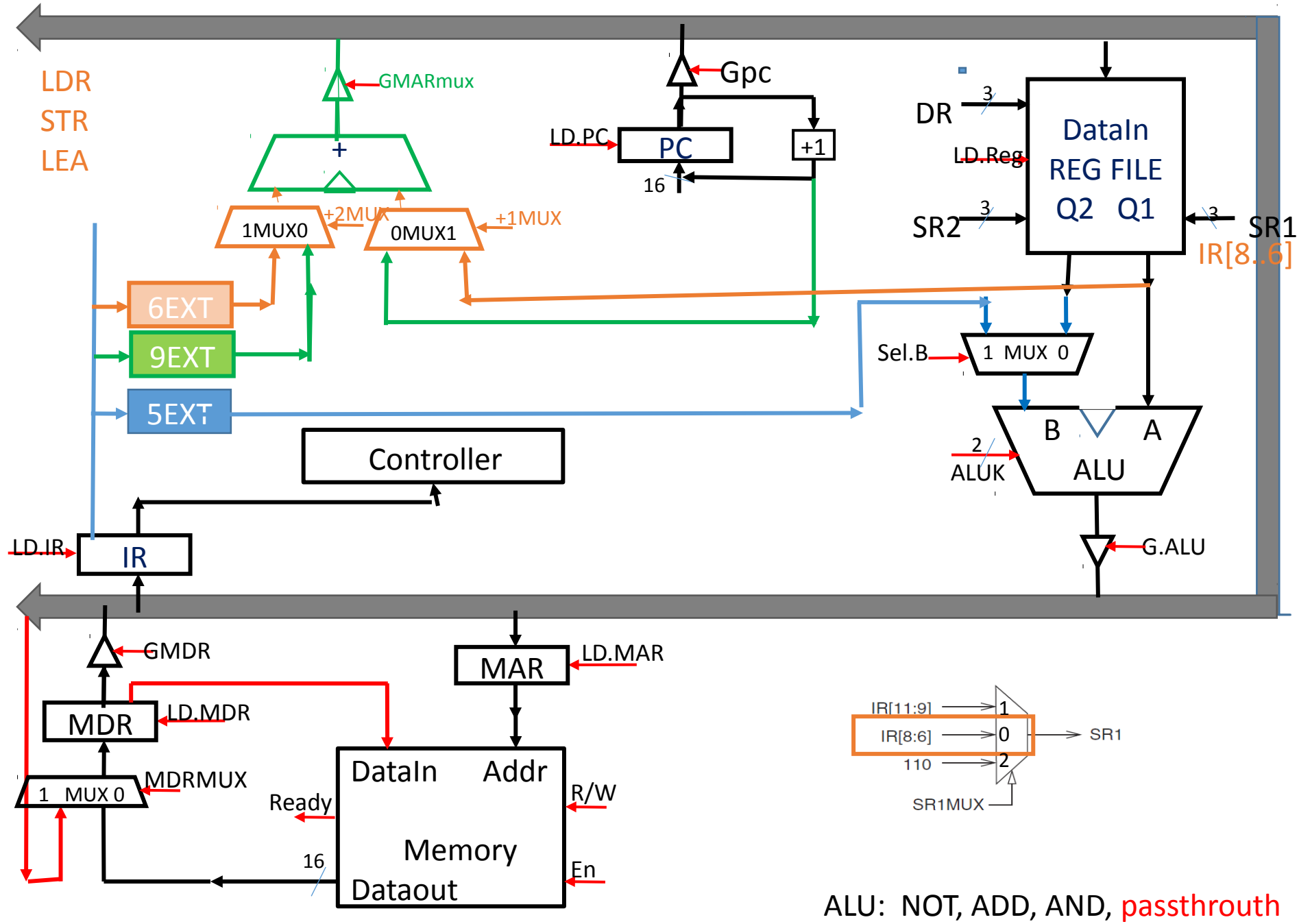
LEA (Immediate)





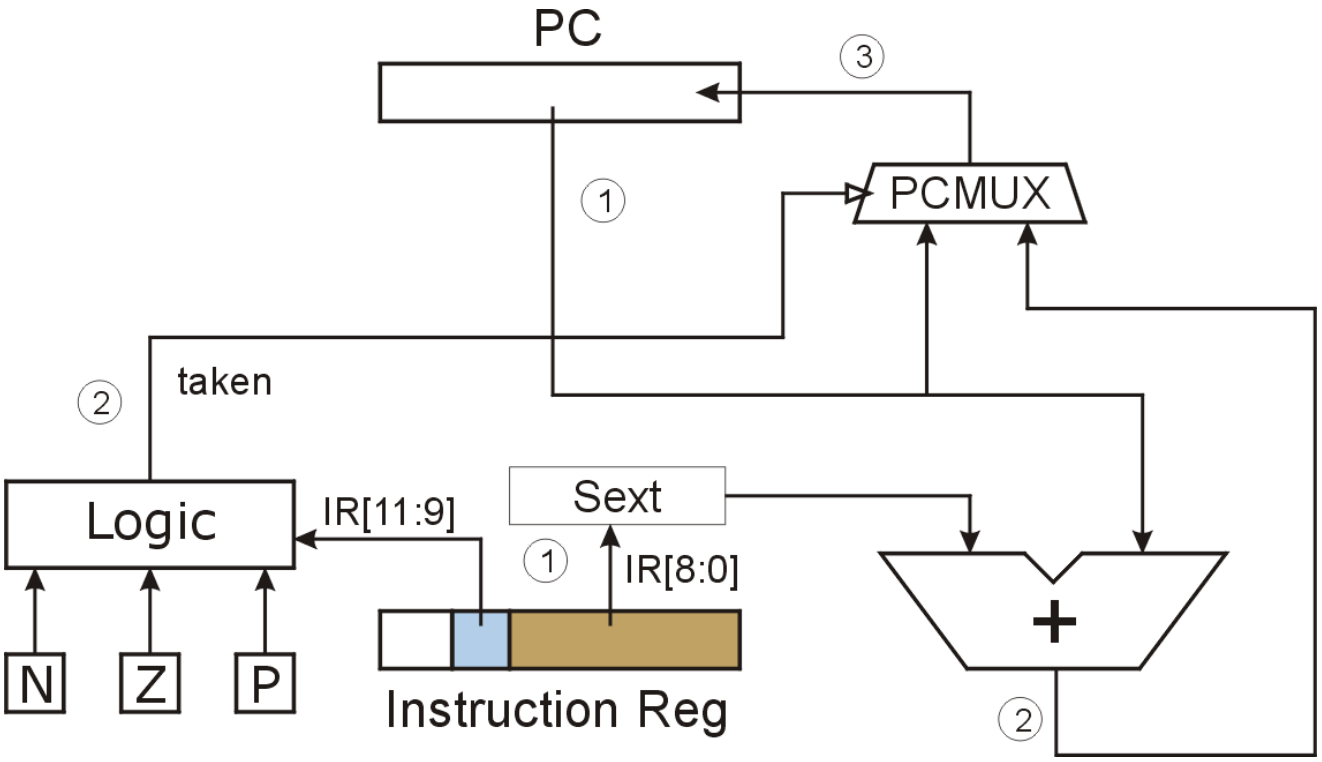
Truth table for Control signals

		Gpc	LD.MAR	LD.PC	R/W	En	LD.MDR	G.MDR	LD.IR	GALU	ALUK	LD.Reg	Sel.B	GmARMUX	SR1MUX	MDRMUx	+1MUX	+2MUX	
Instr. Fetch	S18	1	1	1											0	0			
	S33				0	1	1								0	0			
	S35							1	1						0	0			
Decode	S32														0	0			
NOT	S9									1	NOT	1			0	0			
+/*R	S1									1	+/*	1			0	0			
+/* i	S5									1	+/*	1	1		0	0			
LD/LDI /ST/STI	S2/S3/ S10/s11		1											1	0	0			
LD /LDR /LDI	S25				0	1	1								0	0			
	S27							1				1			0	0			
ST /STR /STI	S23							1		1					1	1			
	S16				1	1													
LDI/STI	S26/S31		1					1											
LDR/STR	S6/s7		1											1	0		1	1	
LEA	S14											1		1			0	0	


LDR
STR
LEA
GMARmux
+
1MUX0
+2MUX
0MUX1
+1MUX
Gpc
LD.PC
PC
+1
16
DR
3
LD.Reg
DataIn
REG FILE
Q2 Q1
SR2
3
3
SR1
IR[8..6]
6EXT
9EXT
5EXT
Sel.B
1 MUX 0
B
A
2
ALUK
ALU
G.ALU
Controller
LD.IR
IR
GMDR
MDR
LD.MDR
MDRMUX
1 MUX 0
MAR
LD.MAR
DataIn
Addr
R/W
Ready
Memory
16
Dataout
En
IR[11:9]
IR[8:6]
110
1
0
2
SR1
SR1MUX
ALU: NOT, ADD, AND, passthrouth

BRn PCoffset ;if(CC == n) PC <- PC + PCoffset

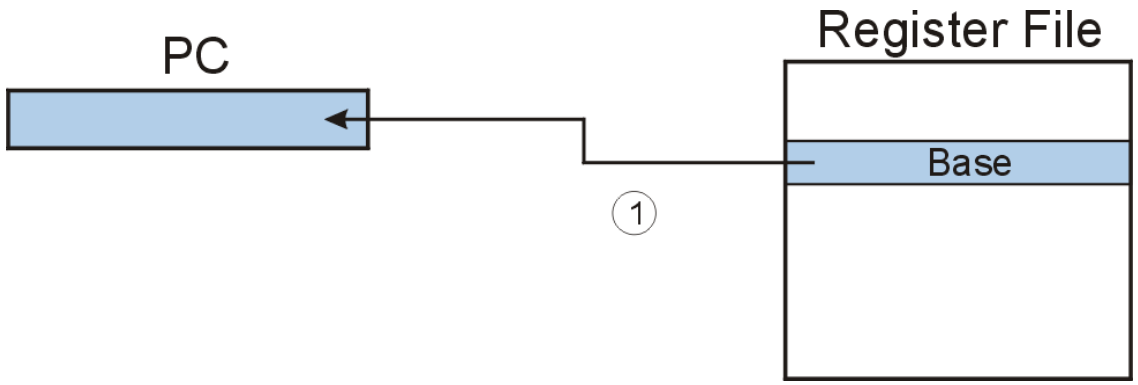
Taken = (N*n + Z*z + P*p)* BR

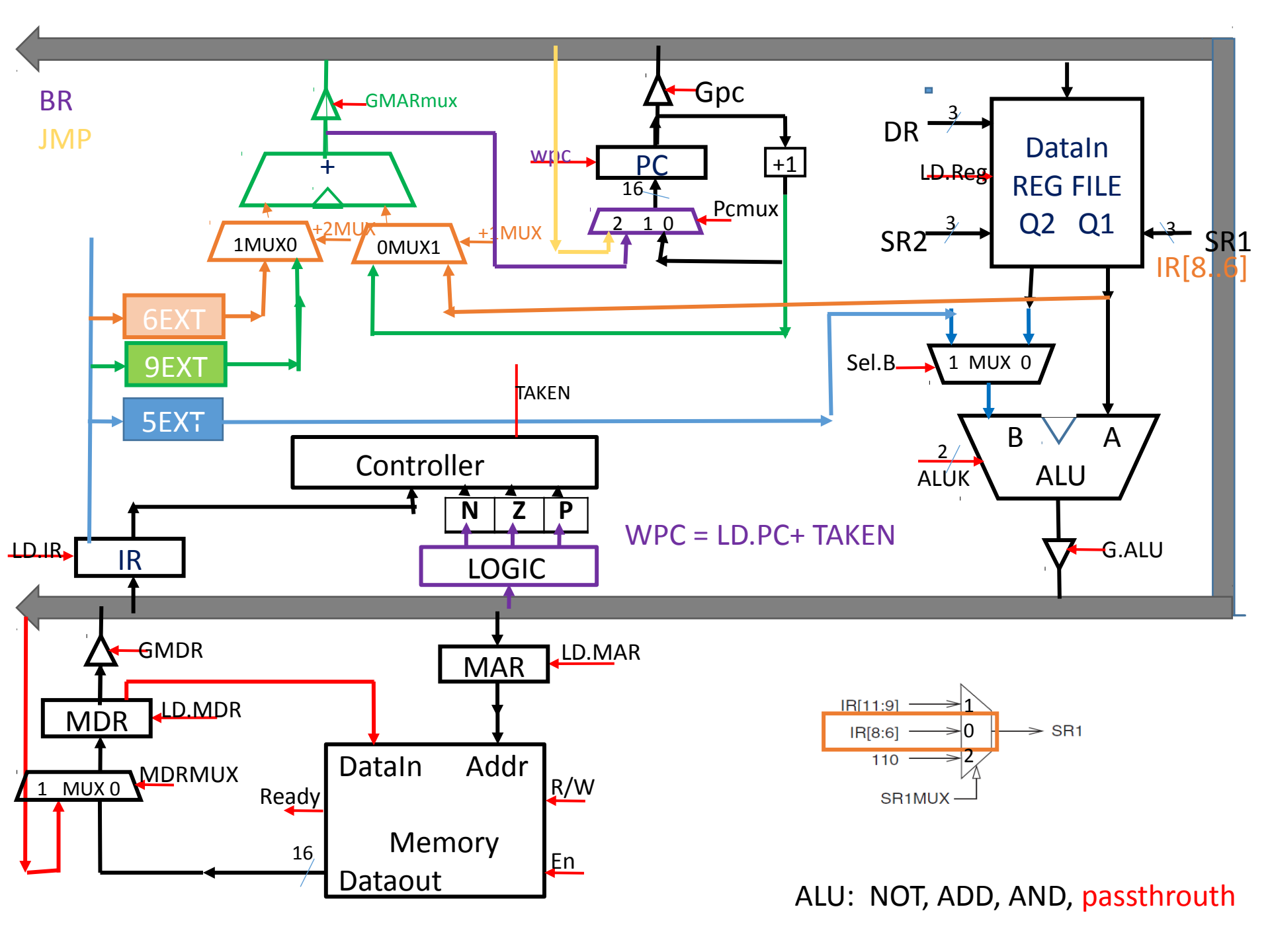


hat happens if bits [11:9] are all zero? All one?

JMP (Register)

- Jump is an unconditional branch -- *always* taken.
  - Target address is the contents of a register.
  - Allows any target address.




BR
JMP
GMARmux
+
+2MUX
1MUX0
0MUX1
+1MUX
wpc
Gpc
PC
16
+1
Pcmux
2 1 0
DR
3
LD.Reg
DataIn
REG FILE
Q2 Q1
SR2
3
3
SR1
IR[8..6]
6EXT
9EXT
5EXT
TAKEN
Sel.B
1 MUX 0
B
A
2
ALUK
ALU
Controller
N Z P
LOGIC
WPC = LD.PC+ TAKEN
G.ALU
LD.IR
IR
GMDR
MAR
LD.MAR
MDR
LD.MDR
MDRMUX
1 MUX 0
DataIn
Addr
R/W
Ready
Memory
16
Dataout
En
IR[11:9]
IR[8:6]
110
1
0
2
SR1
SR1MUX
ALU: NOT, ADD, AND, passthrouth

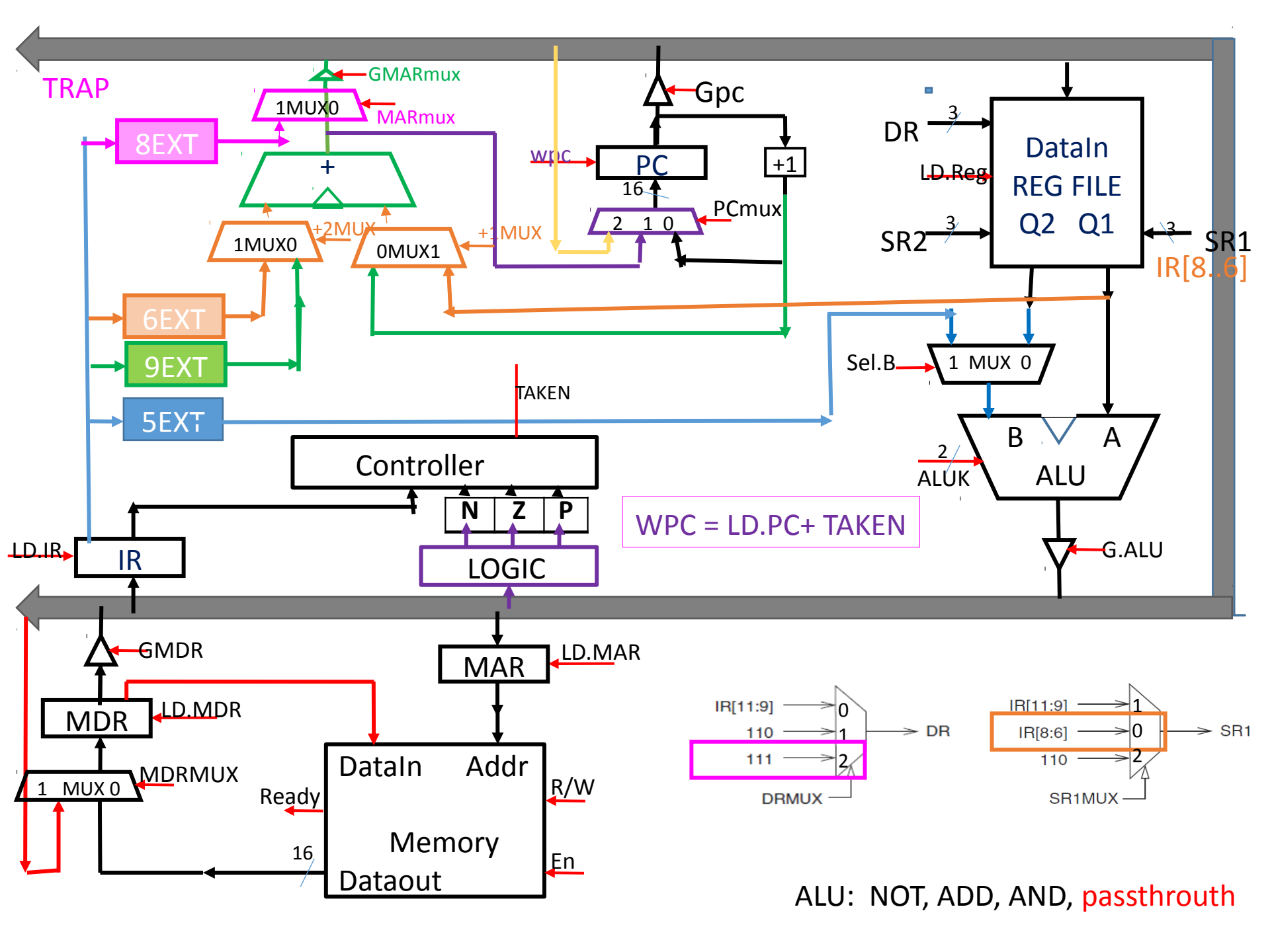
		Gpc	LD.MAR	LD.PC	DRMUX	R/W	En	LD.MDR	G.MDR	LD.IR	GALU	ALUK	LD.Reg	Sel.B	GmARMUX	SR1MUX	MDRMUx	+1MUX	+2MUX	pcMuX	MARMuX
Instr. Fetch	S18	1	1	1												0	0				
	S33					0	1	1								0	0				
	S35								1	1						0	0				
Decode	S32															0	0				
NOT	S9										1	NOT	1			0	0				
+/*R	S1										1	+/*	1			0	0				
+/* i	S5										1	+/*	1	1		0	0				
BR	S22			0									1		1	0		0	0	1	1
JMP	s12			1							1	pass				0				2	2
Trap	S15		1												1						1
	S28	1			2	1	1	1					1				0				
	S30																				

TRAP x21 ; R7←PC; PC <- Memory[x21]

- Calls a service routine, identified by 8-bit "trap vector."

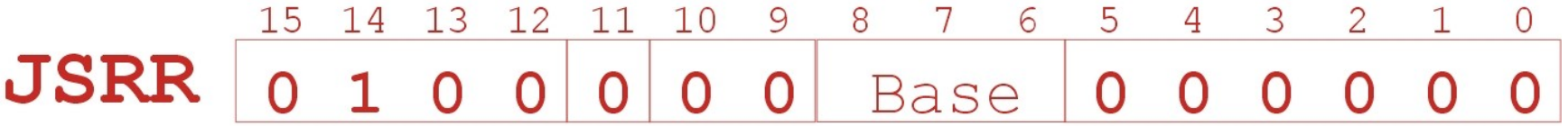
vector	*routine*
x23	**input a character from the keyboard**
x21	**output a character to the monitor**
x25	**halt the program**

- When routine is done,
  PC is set to the instruction following TRAP.
- (We'll talk about how this works later.)


TRAP
GMARmux
1MUX0
MARmux
8EXT
+
wpc
Gpc
PC
+1
16
PCmux
2 1 0
+2MUX
+1MUX
1MUX0
0MUX1
6EXT
9EXT
5EXT
TAKEN
DR
3
LD.Reg
SR2
3
DataIn
REG FILE
Q2 Q1
3
SR1
IR[8..6]
Sel.B
1 MUX 0
B
A
2
ALUK
ALU
G.ALU
Controller
N Z P
LOGIC
WPC = LD.PC+ TAKEN
LD.IR
IR
GMDR
MAR
LD.MAR
MDR
LD.MDR
MDRMUX
1 MUX 0
DataIn
Addr
R/W
Ready
Memory
16
Dataout
En
IR[11:9]
110
111
0
1
2
DR
DRMUX
IR[11:9]
IR[8:6]
110
1
0
2
SR1
SR1MUX
ALU: NOT, ADD, AND, passthrouth

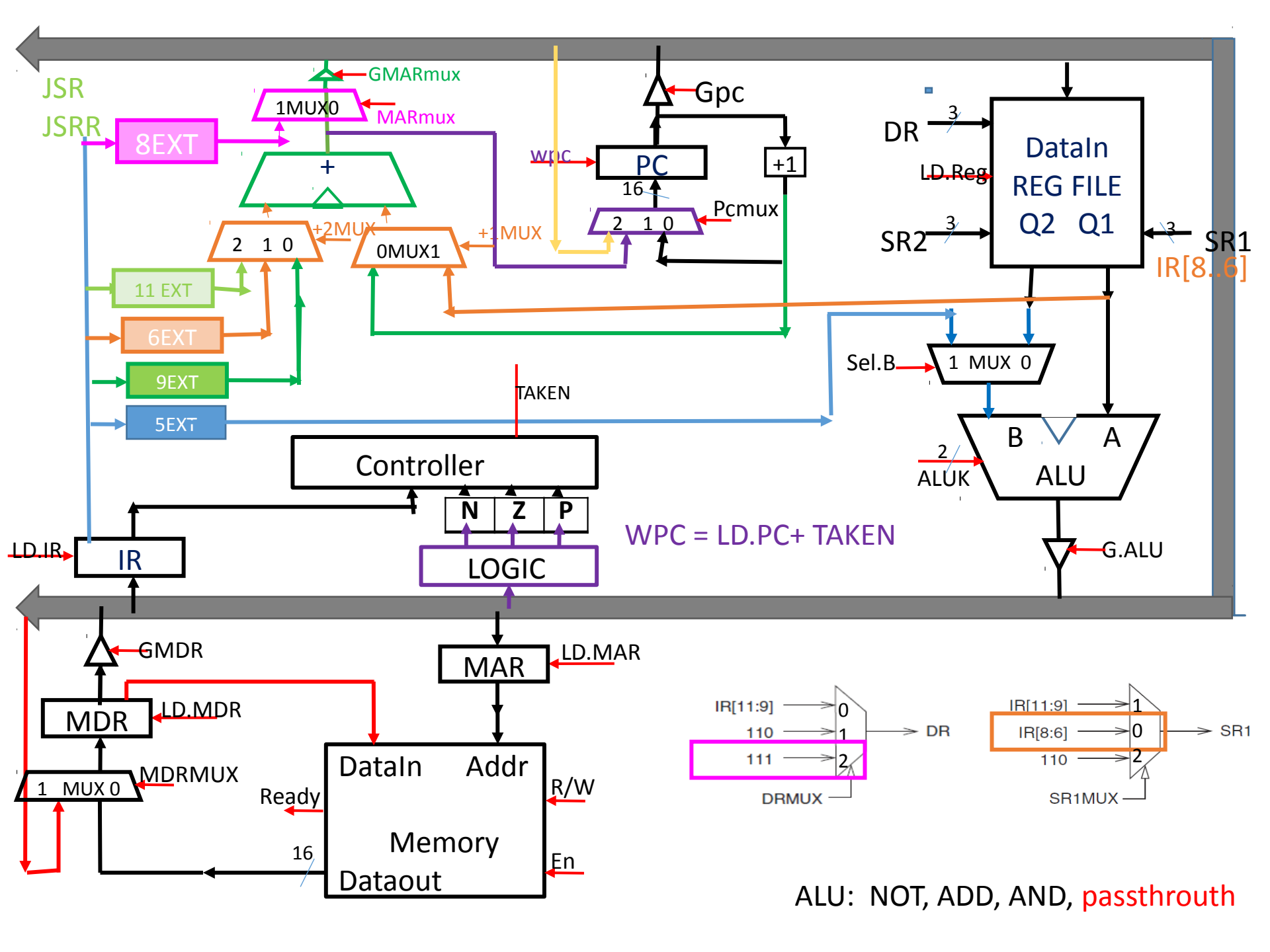
CH5 LC-3

JSR PCoffset11 ;R7 <- PC, PC <- PC + PCoffset



JSR R1 ;R7 <- PC, PC <- R1

		Gpc	LD.MAR	LD.PC	DRMUX	R/W	En	LD.MDR	G.MDR	LD.IR	GALU	ALUK	LD.Reg	Sel.B	GmARMUx	SR1MUX	MDRMUx	+1MUX	+2MUX	pcMuX	MARMuX
Instr. Fetch	S18	1	1	1												0	0				
	S33					0	1	1								0	0				
	S35								1	1						0	0				
Decode	S32															0	0				
NOT	S9										1	NOT	1			0	0				
+/*r	S1										1	+/*	1			0	0				
+/* i	S5										1	+/*	1	1		0	0				
BR	S22			0									1		1	0		0	0	1	1
JMP	s12			1							1	pass				0				2	2
Trap	S15		1												1						1
	S28	1			2	1	1	1					1				0				
	S30																				
JSR	S4	1			2								1								
	S21			1														0	2	1	
JSRR	S20			1							1	pass				0				2	


JSR
JSRR
8EXT
GMARmux
1MUX0
MARmux
+
+2MUX
2 1 0
0MUX1
+1MUX
wpc
Gpc
PC
16
+1
Pcmux
2 1 0
DR
3
LD.Reg
DataIn
REG FILE
Q2 Q1
SR2
3
3
SR1
IR[8..6]
11 EXT
6EXT
9EXT
5EXT
TAKEN
Sel.B
1 MUX 0
B
A
2
ALUK
ALU
Controller
N Z P
LOGIC
WPC = LD.PC+ TAKEN
LD.IR
IR
G.ALU
GMDR
MAR
LD.MAR
MDR
LD.MDR
MDRMUX
1 MUX 0
DataIn
Addr
R/W
Ready
Memory
16
Dataout
En
IR[11:9]
110
111
0
1
2
DR
DRMUX
IR[11:9]
IR[8:6]
110
1
0
2
SR1
SR1MUX
ALU: NOT, ADD, AND, passthrouth