LC3b Architecture

Data Path

The data path includes the following major blocks:

1. Memory
2. Register File
3. ALU, PSR(flags) and sign/zero extenders
4. Program Counter

The interconnections of these blocks and corresponding control signals are shown in the figure:

Description:

Memory:

Memory consists of 256 byte space. Each byte is comprised of 16 bits. The 16 bits are divided into two 8-bits, memH and memL for upper 8 and lower 8 bits respectively.

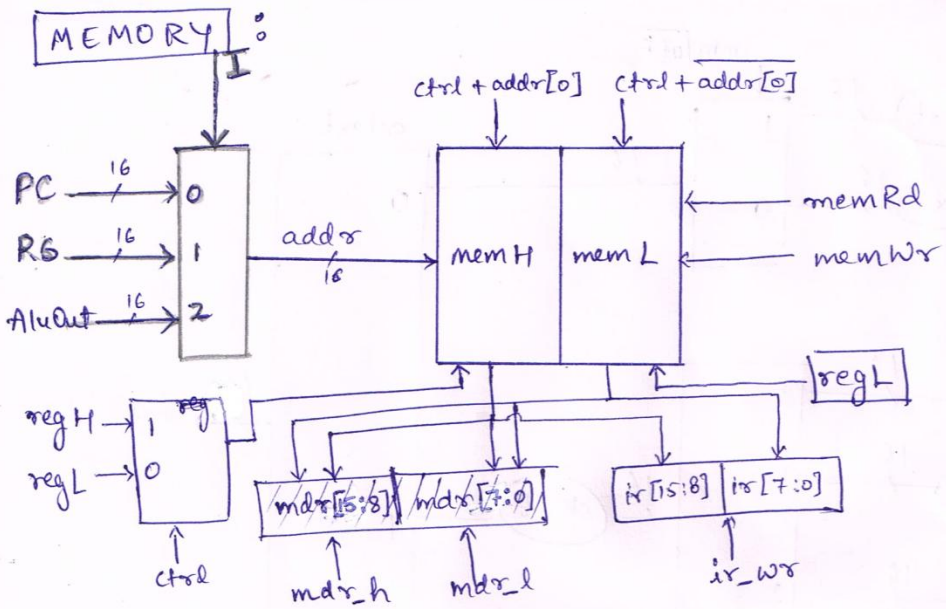
It has a latch I to select the address from PC/R6/AluOut. The address then decides which memory location is accessed. Control bits addr[0] and ctrl decide if memH or memL or both are active. It thus decides if we will access a word (16 bits) or a byte (8 bits).

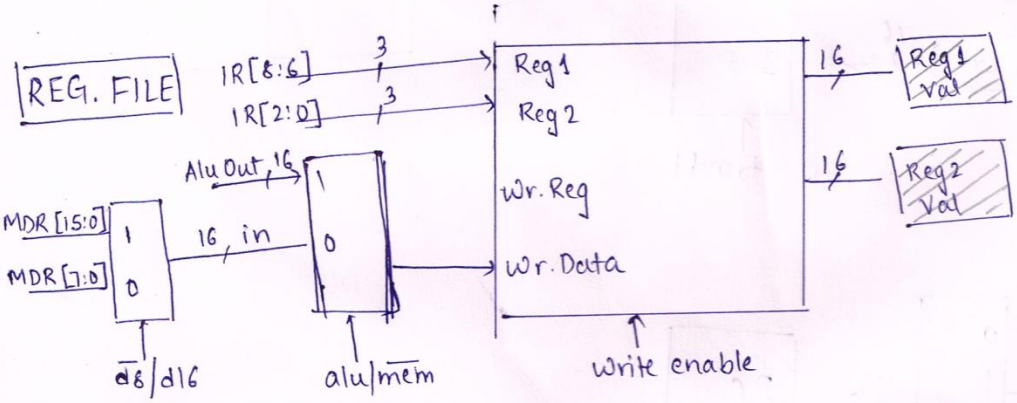
When memWr is enabled, if we are writing a word, data is written from regH and regL into memH and memL respectively. Else, if we are writing a byte, data is written from regL into memH or memL depending upon which one of them is active.

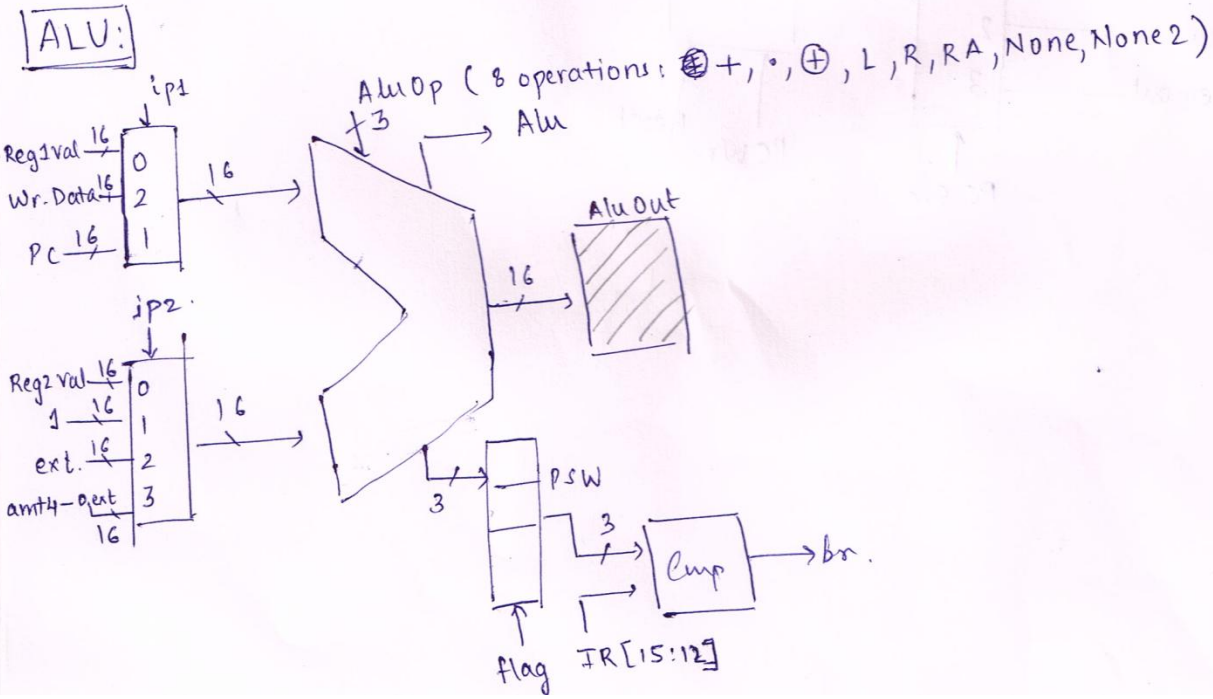
When memRd is enabled, data is read from the corresponding memory location to either ir or mdr depending upon which bit is enabled among ir_wr, mdr_h, mdr_l. If ir_wr is enabled, complete 16 bit data is read from memory to ir register. If one of mdr_h or mdr_l is enabled, byte is read from one of memH or memL register into mdr[15:8] or mdr[7:0]. If both mdr_h and mdr_l are enabled, complete 16 bits word is read from memory to mdr[15:0].

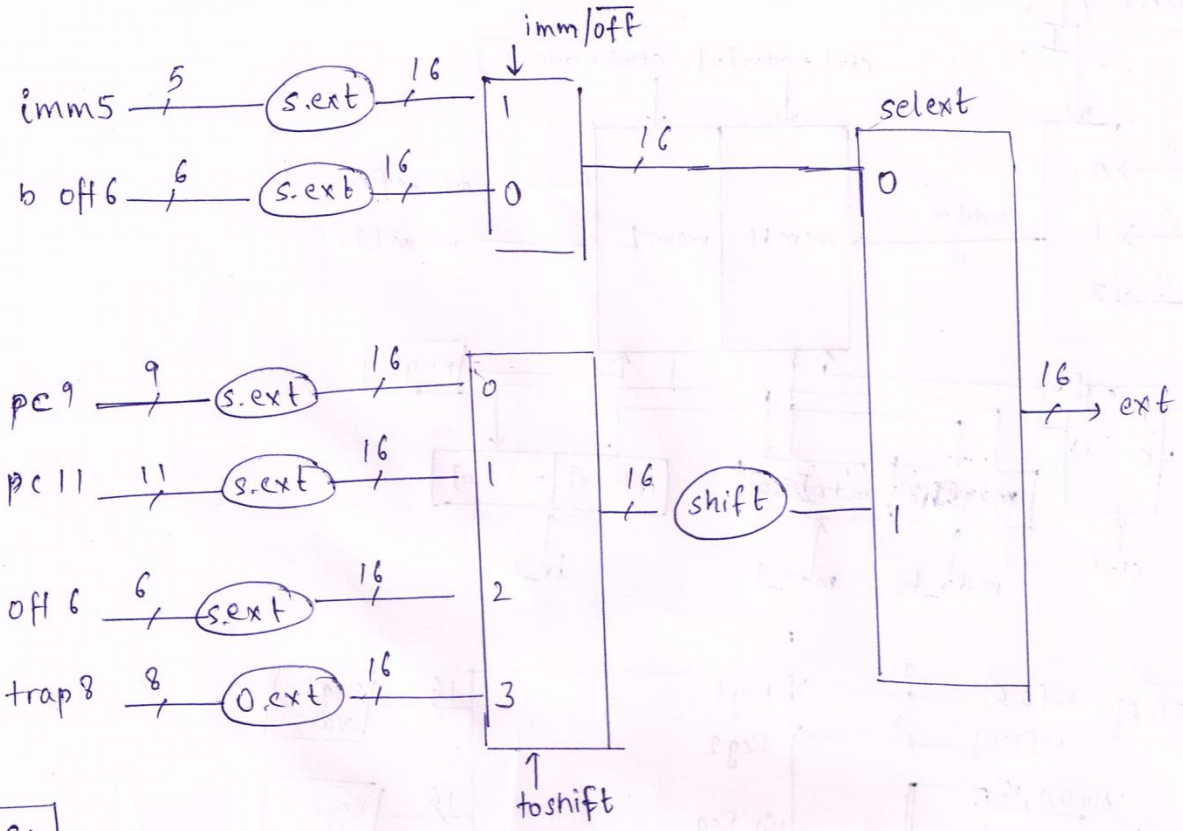
Register File:

Register file consists of 8 16 bit registers,2 multiplexers with control bits d8_d16 and alu_mem which select whether to process 8 bit data or 16 bit data and whether to take data from memory or from alu respectively.Reg1val and Reg2val are two latches which implements reading function from registers according to the value given in IR.The control bit write_en enables writing or loading a value into a particular register the address of which is given as input.

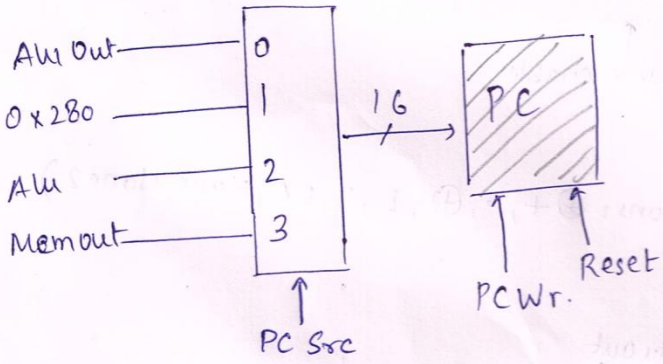

MEMORY
ctrl + addr[0]
ctrl + addr[0]
PC
16
RG
16
AluOut
16
0
1
2
addr
16
memH
memL
memRd
memWr
regH
regL
reg
mdr[15:8]
mdr[7:0]
ir[15:8]
ir[7:0]
ctrl
mdr_h
mdr_l
ir_wr


REG. FILE
IR[8:6]
3
IR[2:0]
3
Reg1
Reg2
Wr. Reg
Wr. Data
16
Reg1 val
16
Reg2 val
AluOut
16
MDR[15:0]
MDR[7:0]
16, in
d8/d16
alu/mem
write enable


ALU:
ip1
ip2
Reg1val
16
Wr. Data
16
PC
16
Reg2 val
16
1
16
ext.
16
amt4-0 ext
16
AluOp (8 operations: +, ·, ⊕, L, R, RA, None, None2)
3
Alu
16
AluOut
16
3
PSW
3
Cmp
br.
flag
IR[15:12]


imm/off
imm5
5
s.ext
16
1
b off6
6
s.ext
16
0
16
selext
0
16
ext
pc9
9
s.ext
16
0
pc11
11
s.ext
16
1
16
shift
1
off6
6
s.ext
16
2
trap8
8
0.ext
16
3
toshift

PC:


ALU out
0x280
ALU
Memout
0
1
2
3
16
PC
PC Wr.
Reset
PC Src

ALU:

ALU performs 8 different operations, controlled by op: Add, And, XOR, Left shift, Right Shift, Right Shift Arithmetic, None1, None2. None operations give output=input. ALU has 2 inputs. None1 makes output=input1 and None2 makes output=input2.

Input 1 is selected from Reg1val, WrData(also a Regfile inputs, 8/16 bit version of MDR), PC, R6 by using a mux controlled by ip1 signal.

Input 2 is selected from Reg2val, 0x02, extender output, zero-extended amount4 by using a mux controlled by ip2 signal.

Extender unit has imm5, boff6, pc9, pc11, off6, trap8 as inputs. These are directly taken from corresponding instructions. Of these imm5, boff6 need only sign extension and pc9, pc11, off6, trap8 need sign/zero extension with left shift. These are controlled by 3 muxes controlled by: imm/off, toshift, sel_ext.

Since alu is combinatorial, its output is also available in the same cycle. This output is called alu(wire). Output of ALU is stored in alu_out latch for further use in other clock cycles.

ALU also controls flags. Flags get updated only when flag signal is set. Flags can be loaded by validating alu_out latch or can be directly from memory(MDR). This is controlled by frm_mem signal.

Program Counter:

Program counter consists of a 16-bit register called pc(latch) and a mux with control line pc_src. Input to the muxes are either Alu out or Alu or Mem out or 0x280h(in case of Interrupts). The value of the register pc gets updated with either of the inputs depending on the pc_src value. Control bit pc_wr enable writing to pc.

Note:

- Same Memory block is used for both instructions and data(**Von Neumann architecture**).
- Clock and reset are external to both data and control path.
- All reset pins of individual blocks have the same name. This is because they are all tied together.

Control Path

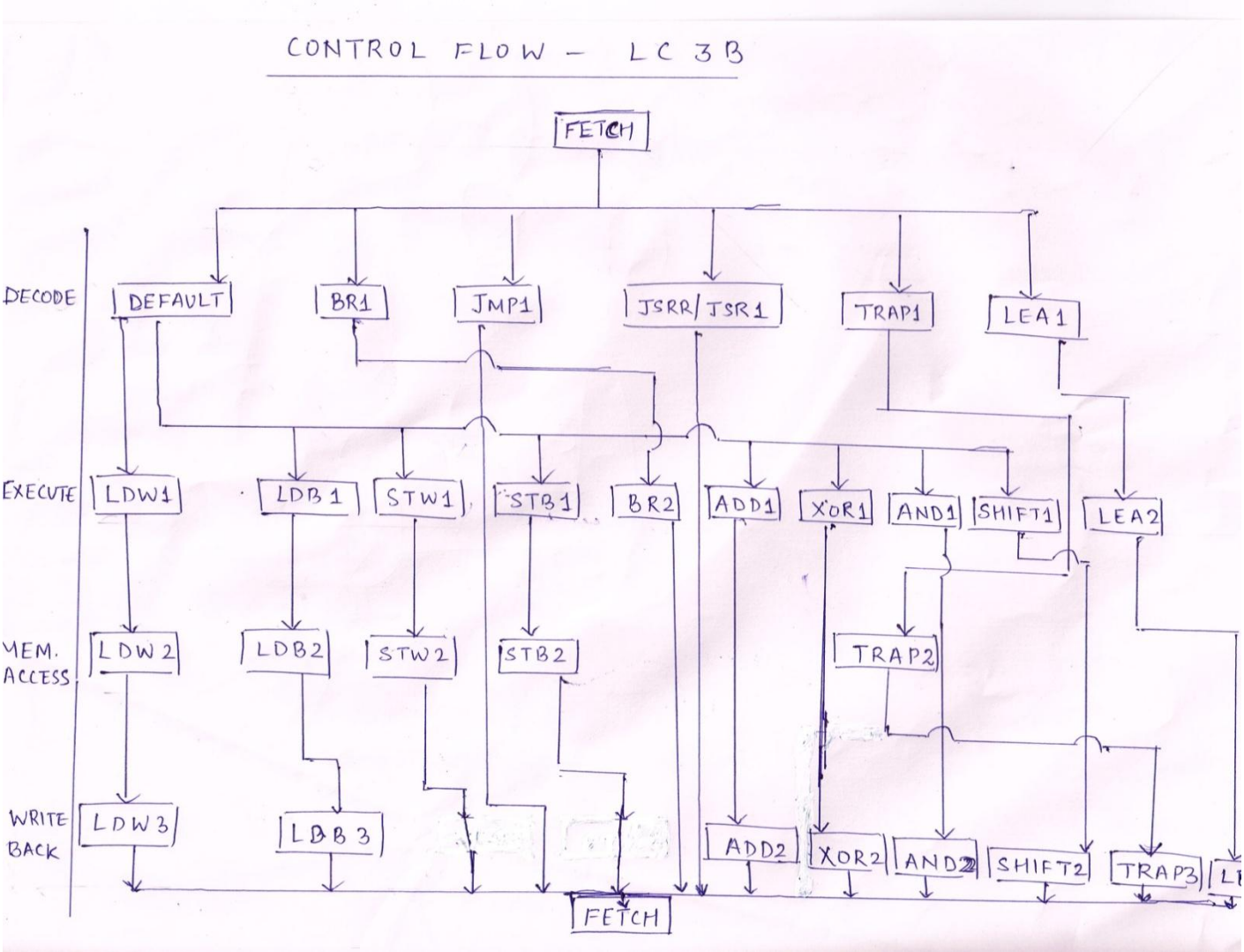
The control path has the following types of states like most RISC architectures:

1. Fetch
2. Decode
3. Execute
4. Memory Access
5. Write Back

Upon reset, reset pins of all the data path blocks are pulled high and the controller enters "Fetch" state in next clock cycle. It issues the corresponding control signals and jumps to next state at positive clock edge and this continues.

The reset operation clears register files, sets PC to start of memory and memory is loaded with the predefined Assembly Language code and corresponding data in specific memory locations.

The control flow is shown in the figure



:

Control signals corresponding to each state are explained below:

Fetch

1. Fetch:

Signals to enable memory read and disable write. Read line is selected to load address from PC. The instruction is written into IR(Instruction Register).

RegFile is disabled.

PC is loaded with PC+2 from ALU.

pc_src=2'b01; //from alu(imm)
pc_wr=1'b1;
	write_en=1'b0;
	reg_reset =1'b0;
		I=0;
		memRd=1;

pc_reset=1'b0;	ip1_sel=2'b01;
	ip2_sel=2'b01;
	op=3'b000;
	flag=0;
	frm_mem=0;
		memWr=0;
		ctrl=1;
		ir_wr=1'b1;
		mdr_l=1'b0;
		mdr_h=1'b0;
		reset =0;

Decode

2. Default:

Read Register1, Register 2 from Register File.
Disable all other blocks and corresponding latches read/write.

pc_wr=0;
pc_reset=0;
	write_en=0;
	
	frm_mem=0;
	flag=0;
		memRd=0;
		memWr=0;
		ir_wr=0;
		mdr_h=0;
		mdr_l=0;

3. BR1:

ALU adds shifted/sign extended pc-offset to pc. Output stays in alu_out latch. Rest all is disabled.

pc_wr=0;
pc_reset=0;
ip2_sel=2'b10;
write_en=0;
	op = 3'b000;
	frm_mem=0;
	flag=0;
	toshift=2'b00;
	sel_ext=1;
	ip1_sel=2'b01;
		memRd=0;
		memWr=0;
		ir_wr=0;
		mdr_h=0;
		mdr_l=0;

4. JMP1:

Register file reads BaseR into Reg1val latch. ALU passes Reg1val latch to PC. PC gets updated from alu in the same cycle.

pc_wr=0;
pc_src=2'b01;
pc_reset=0;
	op = 3'b110;
	frm_mem=0;
	flag=0;
		memRd=0;
		memWr=0;
		ir_wr=0;

	imm_offb=0;
	sel_ext=0;
	ip1_sel=2'b00;
	ip2_sel=2'b10;
		mdr_h=0;
		mdr_l=0;
		
		write_en=0;

5. JSR(R)1:

Write R7 of regfile with PC by selecting appropriate read select line.
In case of JSRR, pass BaseR from Reg1val to PC.
In case of JSR, add left-shifted, sign-extended PCoffset to PC.
The result is written from alu to PC in the same cycle.

JSRR:

op = 3'b000;
frm_mem=0;
flag=0;
imm_offb=0;
sel_ext=0;
ip1_sel=2'b00;
ip2_sel=2'b10;
	write_en=1;
	d8_d16=1;
	reg_reset=0;
	alu_mem=1;
	dr=3'b111;
		memRd=0;
		memWr=0;
		
		pc_wr=1;
		pc_src=2'b10;
		pc_reset=0;

JSR:

op = 3'b000;
frm_mem=0;
flag=0;
toshift=1;
sel_ext=1;
ip1_sel=2'b01;
	write_en=1;
	d8_d16=1;
	reg_reset=0;
	alu_mem=1;
	dr=3'b111;
	ip2_sel=2'b10;
		memRd=0;
		memWr=0;
		
		pc_wr=1;
		pc_src=2'b10;
		pc_reset=0;

6. TRAP1:

Write R7 of Regfile with PC value.
ALU calculates trap vector address by zero extending and left shifting the trap vector bits. It involves only one operand. Hence, ALU performs None-2 operation.

pc_wr=1;
pc_src=2'b10;
pc_reset=0;
	op=3'b111;
	frm_mem=0;
	flag=0;
	toshift=2'd3;
	sel_ext=1;
		memRd=0;
		memWr=0;
		
		write_en=1;
		d8_d16=1;

	ip2_sel=2'd2;	reg_reset=0;
		alu_mem=1;
		dr=3'b111;

7. LEA1:

ALU performs PC + left-shifted, sign-extended PCOffset. Rest all disabled.

sel_ext=1;
ip1_sel=2'b01;
ip2_sel=2'b10;
 	
pc_wr=0;
pc_reset=0;
	write_en=0;
	
	op = 3'b000;
	frm_mem=0;
	flag=1;
	toshift=2'b00;
		memRd=0;
		memWr=0;
		ir_wr=0;
		mdr_h=0;
		mdr_l=0;

Execute

8. LDW1:

Memory address is calculated in ALU. Rest all disabled.

pc_wr=0;
pc_reset=0;
op=3'b000;
toshift=2'b10;
sel_ext=1;
	write_en=0;
	
	ip1_sel=0;
	flag=0;
	ip2_sel=2'b10;
		memRd=0;
		memWr=0;
		ir_wr=0;
		mdr_h=0;
		mdr_l=0;

9. LDB1:

Memory address is calculated in ALU. Rest all disabled.

pc_wr=0;
pc_reset=0;
	ip1_sel=0;
	ip2_sel=2'b10;
	imm_offb=0;
	sel_ext=0;
	flag=0;
	op=3'b000;
		memRd=0;
		memWr=0;
		ir_wr=0;
		mdr_h=0;
		mdr_l=0;
		write_en=0;

10. STW1:

Memory address is calculated in ALU. Rest all disabled.

pc_wr=0;
pc_reset=0;
	write_en=0;
	
	ip1_sel=0;
	flag=0;
	ip2_sel=2'b10;
	op=3'b000;
	toshift=2'b10;
	sel_ext=1;
		memRd=0;
		memWr=0;
		ir_wr=0;
		mdr_h=0;
		mdr_l=0;

11. STB1:

Memory address is calculated in ALU. Rest all disabled.

pc_wr=0;
pc_reset=0;
	ip1_sel=0;
	ip2_sel=2'b10;
	imm_offb=0;
	sel_ext=0;t
	flag=0;
	op=3'b000;
		memRd=0;
		memWr=0;
		ir_wr=0;
		mdr_h=0;
		mdr_l=0;
		
		write_en=0;

12. BR2:

If branch is taken, load PC from alu_out latch, else do nothing. Rest all disabled.

If branch taken:

pc_src=0;
pc_wr=1;
	write_en=0;
	
	flag=0;
	frm_mem=0;
		memRd=0;
		memWr=0;
		ir_wr=0;
		mdr_h=0;
		mdr_l=0;

If branch not taken:

pc_wr=0;
pc_reset=0;
	write_en=0;
	
	flag=0;
	frm_mem=0;
		memRd=0;
		memWr=0;
		ir_wr=0;
		mdr_h=0;
		mdr_l=0;

13. ADD1:

It selects 16 bit register value and adds it to the other 16 bit register value.Memory,PC write,register file and memory access are disabled.

ip1_sel=0;
ip2_sel=0;
op=3'b000;
	pc_wr=0;
	pc_reset=0;
	
	flag=0;
	frm_mem=0;
		memRd=0;
		memWr=0;
		ir_wr=0;
		mdr_h=0;
		mdr_l=0;
		
		write_en=0;

14. XOR1:

It selects 16 bit register value and xors it to the other 16 bit register value.Memory,PC write,register file and memory access are disabled.

ip1_sel=0;
ip2_sel=0;
op=3'b010;
	pc_wr=0;
	pc_reset=0;
	
	flag=0;
	frm_mem=0;
		memRd=0;
		memWr=0;
		ir_wr=0;
		mdr_h=0;
		mdr_l=0;
		
		write_en=0

15. AND1:

It selects 16 bit register value and xors it to the other 16 bit register value.Memory,PC write,register file and memory access are disabled.

ip1_sel=0;
ip2_sel=0;
op=3'b001;
	pc_wr=0;
	pc_reset=0;
	
	flag=0;
	frm_mem=0;
		memRd=0;
		memWr=0;
		ir_wr=0;
		mdr_h=0;
		mdr_l=0;
		
		write_en=0

16. SHIFT1:

It selects 16 bit register value and eithers shifts it either left or right or right arithmetic according to ir[5:4].Memory,PC write,register file and memory access are disabled.

if(ir[5:4]==2'b01)
op=3'b100;
 	
if(ir[5:4]==2'b10)
op=3'b101;
	flag=0;
	frm_mem=0;
	
	ip1_sel=0;
	ip2_sel=0;
	
	if(ir[5:4]==0)
	op=3'b011;
		memRd=0;
		memWr=0;
		ir_wr=0;
		mdr_h=0;
		mdr_l=0;
		write_en=0
		pc_wr=0;
		pc_reset=0;

17. LEA2:

It selects 9 bit sign extended PCoffset value, left shifts it by 1 and add to PC latch value. Rest all disabled.

ip1_sel=1;
ip2_sel=2'b10;
op=3'b111;
ip2_sel=2'd3;
op=3'b100;
op=3'b000;
	write_en=0;
	
	pc_wr=0;
	pc_reset=0;
		memRd=0;
		memWr=0;
		
		ir_wr=0;
		mdr_h=0;
		mdr_l=0;

Memory Access

18. LDW2:

It takes AluOut as address and reads word from that memory location into mdr_h and mdr_l. Rest all disabled.

flag=1'b0;
frm_mem=1'b0;
	write_en=1'b0;
	
	pc_wr=1'b0;
	mdr_l=1'b1;
	mdr_h=1'b1;
	ir_wr=1'b0;
		l=2'b10;
		memRd=1'b1;
		memWr=1'b0;
		ctrl=1'b1;

19. LDB2:

It takes AluOut as address and reads byte from that memory location into mdr_l. Rest all disabled.

flag=1'b0;	write_en=1'b0;	l=2'b10;

frm_mem=1'b0;
mdr_l=1'b1;
mdr_h=1'b0;
ir_wr=1'b0;
	pc_wr=1'b0;	memRd=1'b1;
		memWr=1'b0;
		ctrl=1'b0;

20. STW2:

It takes AluOut as address and writes word in that memory location from regL and regH. Rest all disabled.

flag=1'b0;
frm_mem=1'b0;
	write_en=1'b0;
	
	pc_wr=1'b0;
	mdr_h=1'b0;
	ir_wr=1'b0;
		I=2'b10;
		memRd=1'b0;
		memWr=1'b1;
		ctrl=1'b1;
		mdr_l=1'b0;

21. STB2:

It takes AluOut as address and writes byte in that memory location from regL. Rest all disabled.

flag=1'b0;
frm_mem=1'b0;
mdr_h=1'b0;
ir_wr=1'b0;
	write_en=1'b0;
	
	pc_wr=1'b0;
	ctrl=1'b0;
		I=2'b10;
		memRd=1'b0;
		memWr=1'b1;
		mdr_l=1'b0;

22. TRAP2:

It takes AluOut as address and reads from that memory location to mdr_l and mdr_h. Rest all disabled.

ctrl=1'b1;
mdr_l=1'b1;
mdr_h=1'b1;
ir_wr=1'b0;
	write_en=1'b0;
	
	pc_wr=1'b0;
		I=2'b10;
		memRd=1'b1;
		memWr=1'b0;
		flag=1'b0;
		frm_mem=1'b0;

Write Back

23. LDW3:

16-bit mdr is written into wrdata and then Right arithmetic performed over it. Rest all disabled.

ip1_sel=2'd2; op=3'd6;	pc_wr=0; pc_reset=0;	memRd=0; memWr=0;

frm_mem=0;
flag=1;
	reg_reset=0;
	d8_d16=1;
	alu_mem=0;
	dr=ir[11:9];
		ir_wr=0;
		mdr_h=0;
		mdr_l=0;
		
		write_en=1;

24. LDB3:

8-bit mdr is written into wrdata and then taken into ALU without any operation. Rest all disabled.

ip1_sel=2'd2;
op=3'd7;
frm_mem=0;
flag=1;
pc_wr=0;
pc_reset=0;
	reg_reset=0;
	d8_d16=0;
	alu_mem=0;
	dr=ir[11:9];
	
	write_en=1;
		memRd=0;
		memWr=0;
		ir_wr=0;
		mdr_h=0;
		mdr_l=0;

25. ALU2 (ADD2 / XOR2 / AND2 / SHIFT2):

AluOut is written into wrdata and then taken into ALU without any operation. Rest all disabled.

ip1_sel=2'd2;
op=3'd7;
frm_mem=0;
flag=1;
	reg_reset=0;
	alu_mem=1;
	dr=ir[11:9];
	
	pc_wr=0;
	pc_reset=0;
		memRd=0;
		memWr=0;
		ir_wr=0;
		mdr_h=0;
		mdr_l=0;
		write_en=1;

26. TRAP3:

It writes from memout to PC. Rest all are disabled.

pc_wr=1;
pc_reset=0;
pc_src=2'd3;
	write_en=0;
	
	frm_mem=0;
	flag=0;
		memRd=0;
		memWr=0;
		ir_wr=0;
		mdr_h=0;
		mdr_l=0;

27. LEA3:

16-bit mdr is written into wrdata and then taken into ALU with no operation. Rest all disabled.

ip1_sel=2'd2;
op=3'd7;
 	
frm_mem=0;
flag=1;
	reg_reset=0;
	d8_d16=1;
	alu_mem=0;
	dr=ir[11:9];
	
	pc_wr=0;
	pc_reset=0;
		memRd=0;
		memWr=0;
		ir_wr=0;
		mdr_h=0;
		mdr_l=0;
		
		write_en=1;

Control States followed by different instructions

Write Back	Memory Access	Execute	Decode	Fetch	Instruction
ALU2		ADD1	Default	Fetch	**ADD**
ALU2		AND1	Default	Fetch	**AND**
		BR2	BR1	Fetch	**BR**
			JMP1	Fetch	**JMP**
			JSR(R)1	Fetch	**JSR(R)**
LDB3	LDB2	LDB1	Default	Fetch	**LDB**
LDB3	LDB2	LDB1	Default	Fetch	**LDW**
LEA3		LEA2	LEA1	Fetch	**LEA**
SHIFT2		SHIFT1	Default	Fetch	**SHF**
	STB2	STB1	Default	Fetch	**STB**
	STW2	STW1	Default	Fetch	**STW**
TRAP3	TRAP2		TRAP1	Fetch	**TRAP**
XOR2		XOR1	Default	Fetch	**XOR**