8. [Bonus] Mystery Instruction [60 points]

A pesky engineer implemented a mystery instruction on the LC-3b. It is your job to determine what the instruction does. The mystery instruction is encoded as:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1010				DestR			SR1			0	0	0	SR2		

The instruction is only defined if the value of SR2 is greater than the value of SR1.

The modifications we make to the LC-3b datapath and the microsequencer are highlighted in the attached figures (see the four pages at the end of this question). We also provide the original LC-3b state diagram, in case you need it. (As a reminder, the selection logic for SR2MUX is determined internally based on the instruction.)

The additional control signals are

LD_TEMP1/1: NO, YES

LD_TEMP2/1: NO, YES

GateTEMP3/1: NO, LOAD

Reg_IN_MUX/1: BUS, Mystery2 – (Assume BUS is asserted if this signal is not specified)

Mystery_MUX/2: SR2MUX, PASS_1 (outputs value 1), PASS_0 (outputs value 0)

Additional Signals for ALUK: PASS_B (outputs the value from input B), SUB (A-B)

Also note that both of DRMUX and SR1MUX can now choose DR, SR1, and SR2

COND/4:
$COND_{0000}$;Unconditional
$COND_{0001}$;Memory Ready
$COND_{0010}$;Branch
$COND_{0011}$;Addressing mode
$COND_{0100}$;Mystery 1
$COND_{1000}$;Mystery 2

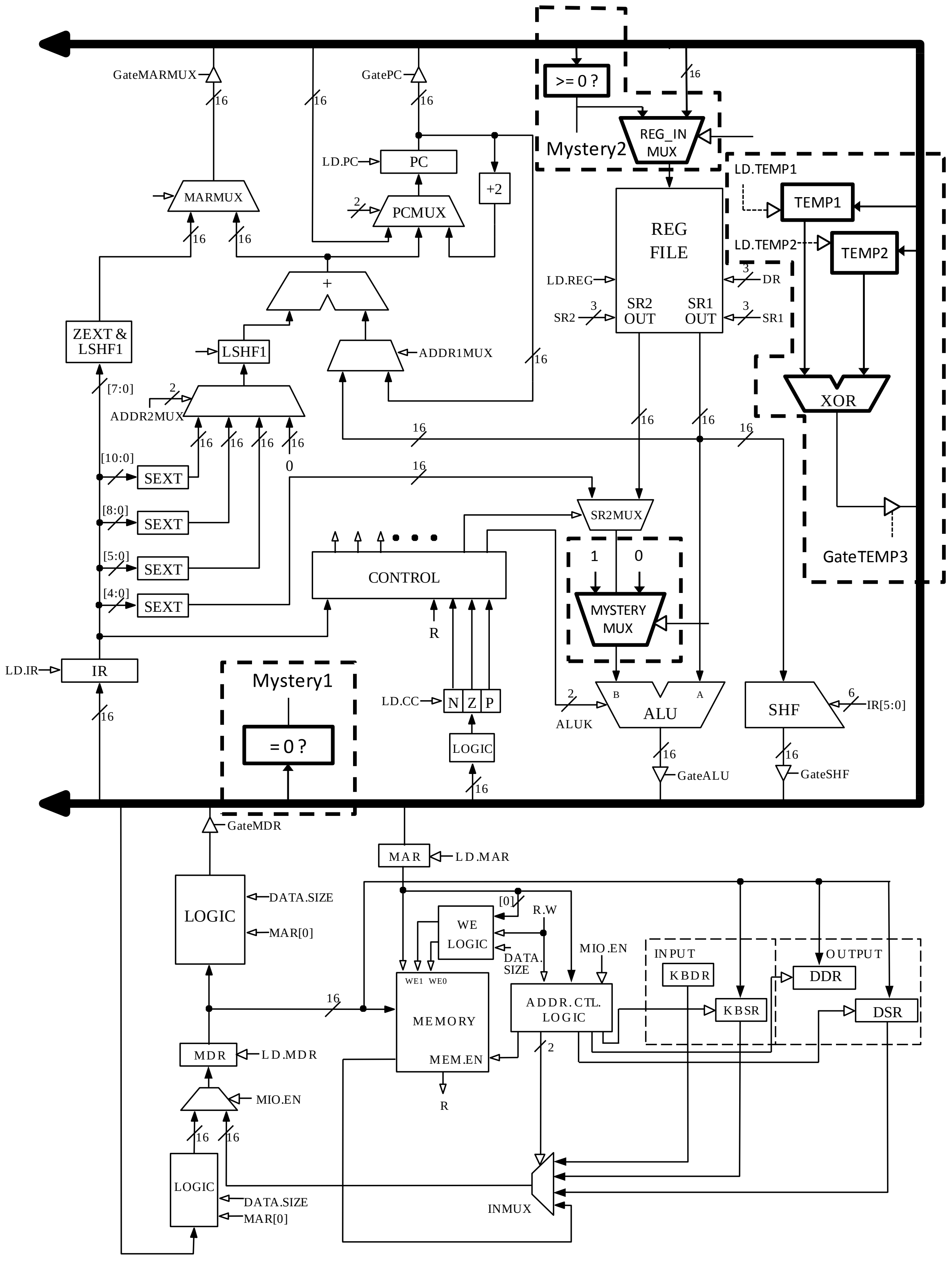
The microcode for the instruction is given in the table below.

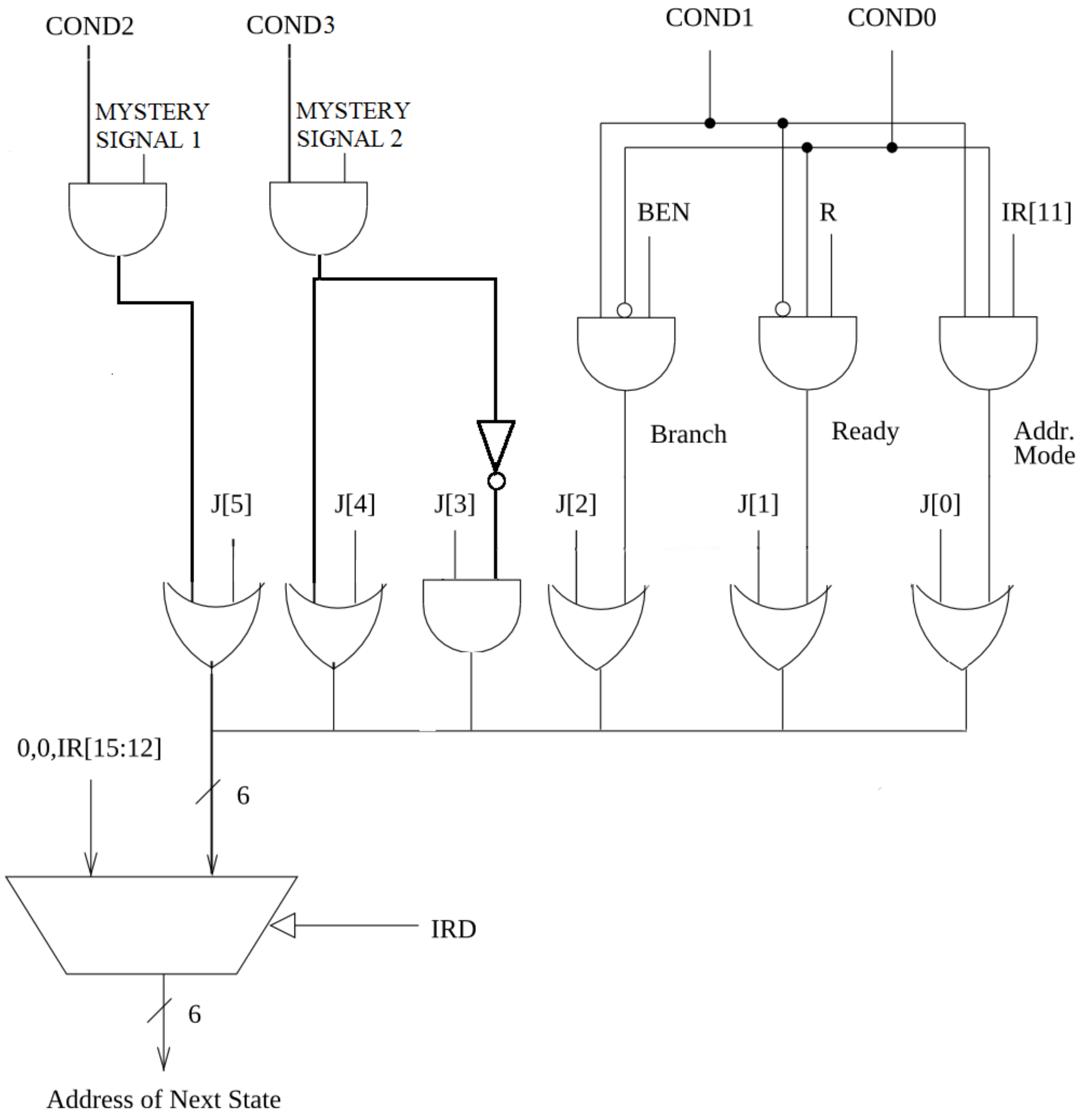
State	Cond	J	Asserted Signals
001010 (10)	COND_{0000}	001011	LD.REG, DRMUX = DR(IR [11:9]), GateALU, ALUK = PASS_B, MYSTERY_MUX = PASS_0
001011 (11)	COND_{0000}	110001	LD.MAR, SR1MUX = SR1(IR[8:6]), ADDR1MUX = SR1OUT, ADDR2MUX = 0, MARMUX = ADDER, GateMARMUX
110001 (49)	COND_{0001}	110001	LD.MDR, MIO.EN, DATA.SIZE=BYTE, R.W = R
110011 (51)	COND_{0000}	100100	GateMDR, LD.TEMP1, DATA.SIZE=BYTE
100100 (36)	COND_{0000}	100101	LD.MAR, SR1MUX = SR2(IR[2:0]), ADDR1MUX = SR1OUT, ADDR2MUX = 0, MARMUX = ADDER, GateMARMUX
100101 (37)	COND_{0001}	100101	LD.MDR, MIO.EN, DATA.SIZE=BYTE, R.W = R
100111 (39)	COND_{0000}	101000	GateMDR, LD.TEMP2, DATA.SIZE=BYTE
101000 (40)	COND_{0100}	010010	GateTEMP3
110010 (50)	COND_{0000}	101001	LD.REG, DRMUX = SR1(IR[8:6]), GateALU, ALUK = ADD, SR1MUX = SR1(IR[8:6]), MYSTERY_MUX = PASS_1
101001 (41)	COND_{0000}	101010	LD.REG, DRMUX = SR2(IR[2:0]), GateALU, ALUK = SUB, SR1MUX = SR2 (IR[2:0]), MYSTERY_MUX = PASS_1
101010 (42)	COND_{1000}	001011	LD.REG, DRMUX = DR (IR[11:9]), Reg_IN_MUX = MYSTERY2, GateALU, ALUK = SUB, SR1MUX = SR1(IR[8:6]), MYSTERY_MUX = SR2MUX

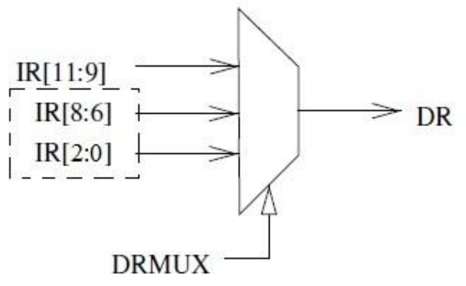
Describe what this instruction does. Show your work for partial credit.

This instruction checks if the given string is a palindrome.

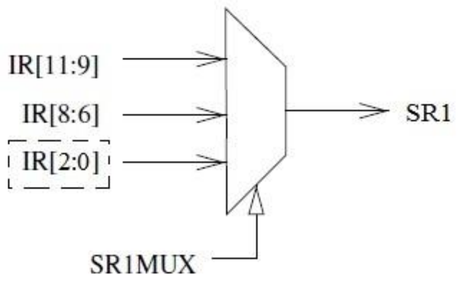
```
Code:
(char * sr1, *sr2;)
destR = 0;
while(sr1 < sr2){
    if (mem[sr1] != mem[sr2])
        return(fetch next instruction)
    sr1++;
    sr2--;
}
destR = 1;
return(fetch next instruction)
```


GateMARMUX
GatePC
16
MARMUX
PC
LD.PC
PCMUX
+2
+
LSHF1
ADDR1MUX
ADDR2MUX
ZEXT & LSHF1
[7:0]
[10:0]
[8:0]
[5:0]
[4:0]
SEXT
0
CONTROL
R
IR
LD.IR
>= 0 ?
Mystery2
REG_IN MUX
REG FILE
LD.REG
SR2
SR2 OUT
SR1 OUT
DR
SR1
LD.TEMP1
TEMP1
LD.TEMP2
TEMP2
XOR
GateTEMP3
SR2MUX
1
0
MYSTERY MUX
ALU
ALUK
SHF
IR[5:0]
GateALU
GateSHF
LD.CC
N
Z
P
LOGIC
Mystery1
= 0 ?
GateMDR
MAR
LD.MAR
LOGIC
DATA.SIZE
MAR[0]
WE LOGIC
R.W
[0]
MIO.EN
MEMORY
WE1 WE0
MEM.EN
ADDR.CTL. LOGIC
INPUT
KBDR
KBSR
OUTPUT
DDR
DSR
MDR
LD.MDR
INMUX

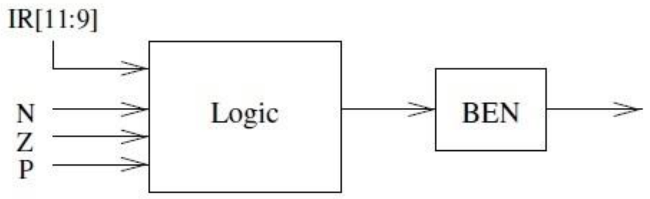

COND2
COND3
COND1
COND0
MYSTERY SIGNAL 1
MYSTERY SIGNAL 2
BEN
R
IR[11]
Branch
Ready
Addr. Mode
J[5]
J[4]
J[3]
J[2]
J[1]
J[0]
0,0,IR[15:12]
6
IRD
6
Address of Next State


IR[11:9]
IR[8:6]
IR[2:0]
DR
DRMUX

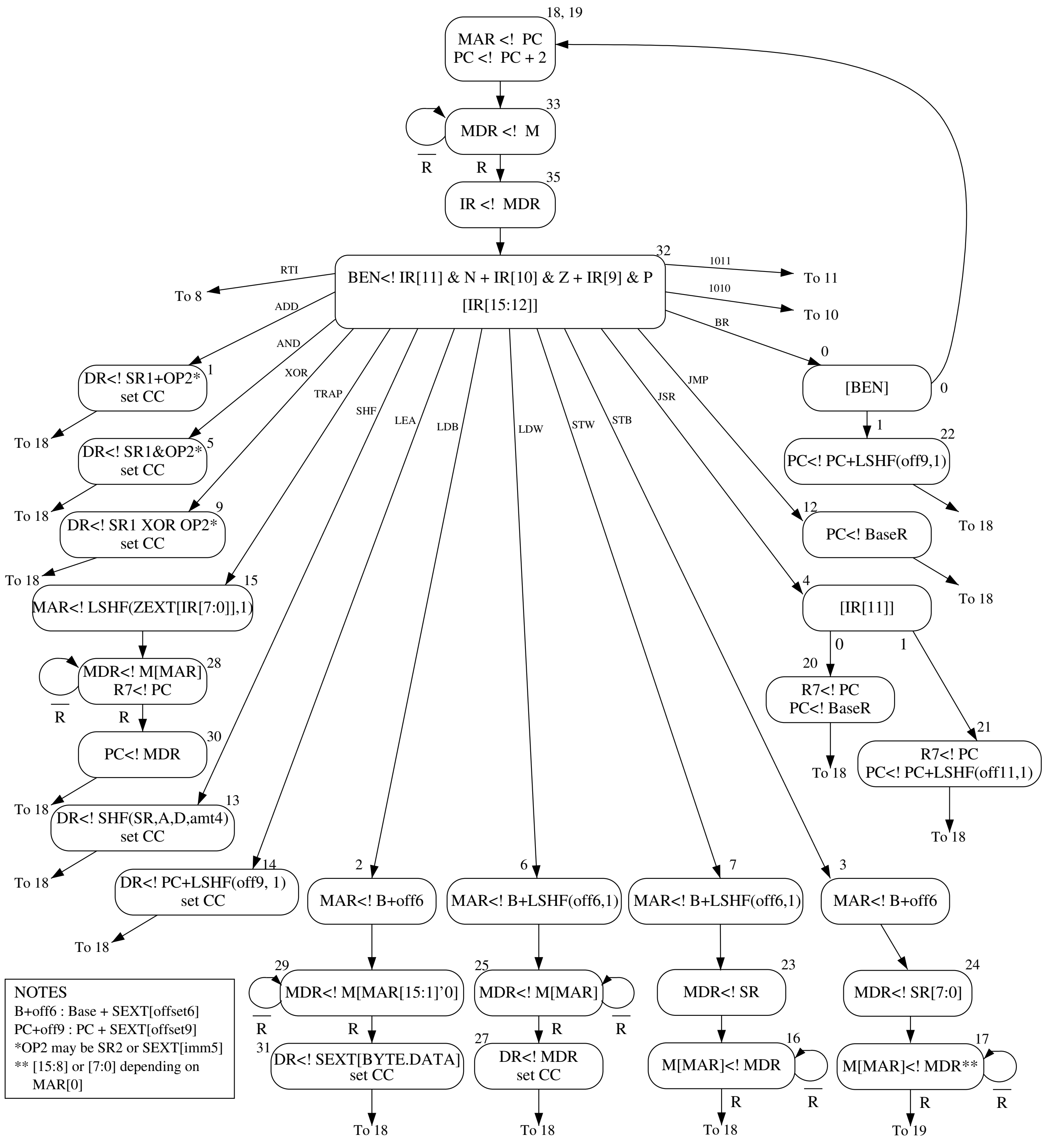
(a)


IR[11:9]
IR[8:6]
IR[2:0]
SR1
SR1MUX

(b)


IR[11:9]
N
Z
P
Logic
BEN

(c)


18, 19
MAR <! PC
PC <! PC + 2
33
MDR <! M
R
R
35
IR <! MDR
32
BEN<! IR[11] & N + IR[10] & Z + IR[9] & P
[IR[15:12]]
RTI
To 8
1011
To 11
1010
To 10
BR
ADD
AND
XOR
TRAP
SHF
LEA
LDB
LDW
STW
STB
JSR
JMP
0
[BEN]
0
1
22
PC<! PC+LSHF(off9,1)
To 18
1
DR<! SR1+OP2*
set CC
To 18
5
DR<! SR1&OP2*
set CC
To 18
9
DR<! SR1 XOR OP2*
set CC
To 18
15
MAR<! LSHF(ZEXT[IR[7:0]],1)
28
MDR<! M[MAR]
R7<! PC
R
R
30
PC<! MDR
To 18
13
DR<! SHF(SR,A,D,amt4)
set CC
To 18
14
DR<! PC+LSHF(off9, 1)
set CC
To 18
12
PC<! BaseR
To 18
4
[IR[11]]
0
1
20
R7<! PC
PC<! BaseR
To 18
21
R7<! PC
PC<! PC+LSHF(off11,1)
To 18
2
MAR<! B+off6
29
MDR<! M[MAR[15:1]'0]
R
R
31
DR<! SEXT[BYTE.DATA]
set CC
To 18
6
MAR<! B+LSHF(off6,1)
25
MDR<! M[MAR]
R
R
27
DR<! MDR
set CC
To 18
7
MAR<! B+LSHF(off6,1)
23
MDR<! SR
16
M[MAR]<! MDR
R
R
To 18
3
MAR<! B+off6
24
MDR<! SR[7:0]
17
M[MAR]<! MDR**
R
R
To 19
NOTES
B+off6 : Base + SEXT[offset6]
PC+off9 : PC + SEXT[offset9]
*OP2 may be SR2 or SEXT[imm5]
** [15:8] or [7:0] depending on
MAR[0]

Stratchpad

VLIW Instruction	**ALU**	**MU**	**FPU**
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			
16			
17			
18			
19			
20			
21			
22			
23			
24			
25			

VLIW Instruction	**ALU**	**MU**	**FPU**
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			
16			
17			
18			
19			
20			
21			
22			
23			
24			
25			

VLIW Instruction	**ALU**	**MU**	**FPU**
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			
16			
17			
18			
19			
20			
21			
22			
23			
24			
25			