8. [Bonus] Mystery Instruction Strikes Back [50 points]

That pesky engineer implemented yet another 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				DR			SR1			0	0	0	0	0	0

The modifications we make to the LC-3b datapath and the microsequencer are highlighted in the attached figures (see the next three pages after the question). We also provide the original LC-3b state diagram, in case you need it.

In this instruction, we specify `SR2OUT` to always output `REG[SR1]`, and `SR2MUX` to output value from the `REGFILE`. Each register has a width of 16 bits.

The additional control signals are:

GateTEMP1/1: NO, YES

GateTEMP2/1: NO, YES

LD.TEMP1/1: NO, LOAD

LD.TEMP2/1: NO, LOAD

ALUK/3: OR1 (`A|0x1`), XOR (`A^B`), LSHF1 (`A<<1`), PASSA, PASS0 (Pass value 0), PASS16 (Pass value 16)

Reg_IN_MUX/2: BUS (passes value from BUS), EQ0 (passes the value from the `==0?` comparator). BUS is asserted if this signal is not specified.

COND/4:
$COND_{0000}$;Unconditional
$COND_{0001}$;Memory Ready
$COND_{0010}$;Branch
$COND_{0011}$;Addressing mode
$COND_{0100}$;Mystery 1
$COND_{1000}$;Mystery 2 (which is set based on the 0th bit of TEMP1)

The microcode for the instruction is given in the table on the next page.

Initials:

State	Cond	J	Asserted Signals
001010 (10)	$COND_{0000}$	001011	ALUK = PASS0, GateALU, LD.REG, DRMUX = DR (IR[11:9])
001011 (11)	$COND_{0000}$	101000	ALUK = PASSA, GateALU, LD.TEMP1, SR1MUX = SR1 (IR[8:6])
101000 (40)	$COND_{0000}$	100101	ALUK = PASS16, GateALU, LD.TEMP2
100101 (37)	$COND_{1000}$	101101	ALUK = LSHF1, GateALU, LD.REG, SR1MUX = DR, DRMUX = DR (IR[11:9])
111101 (61)	$COND_{0000}$	101101	ALUK = OR1, GateALU, LD.REG, SR1MUX = DR, DRMUX = DR (IR[11:9])
101101 (45)	$COND_{0000}$	111111	GateTEMP1, LD.TEMP1
111111 (63)	$COND_{0100}$	100101	GateTEMP2, LD.TEMP2
110101 (53)	$COND_{0000}$	010010	GateALU, ALUK = XOR, SR1MUX = DR (IR[11:9]) LD.REG, DRMUX = DR (IR[11:9]), Reg_IN_MUX = EQ0

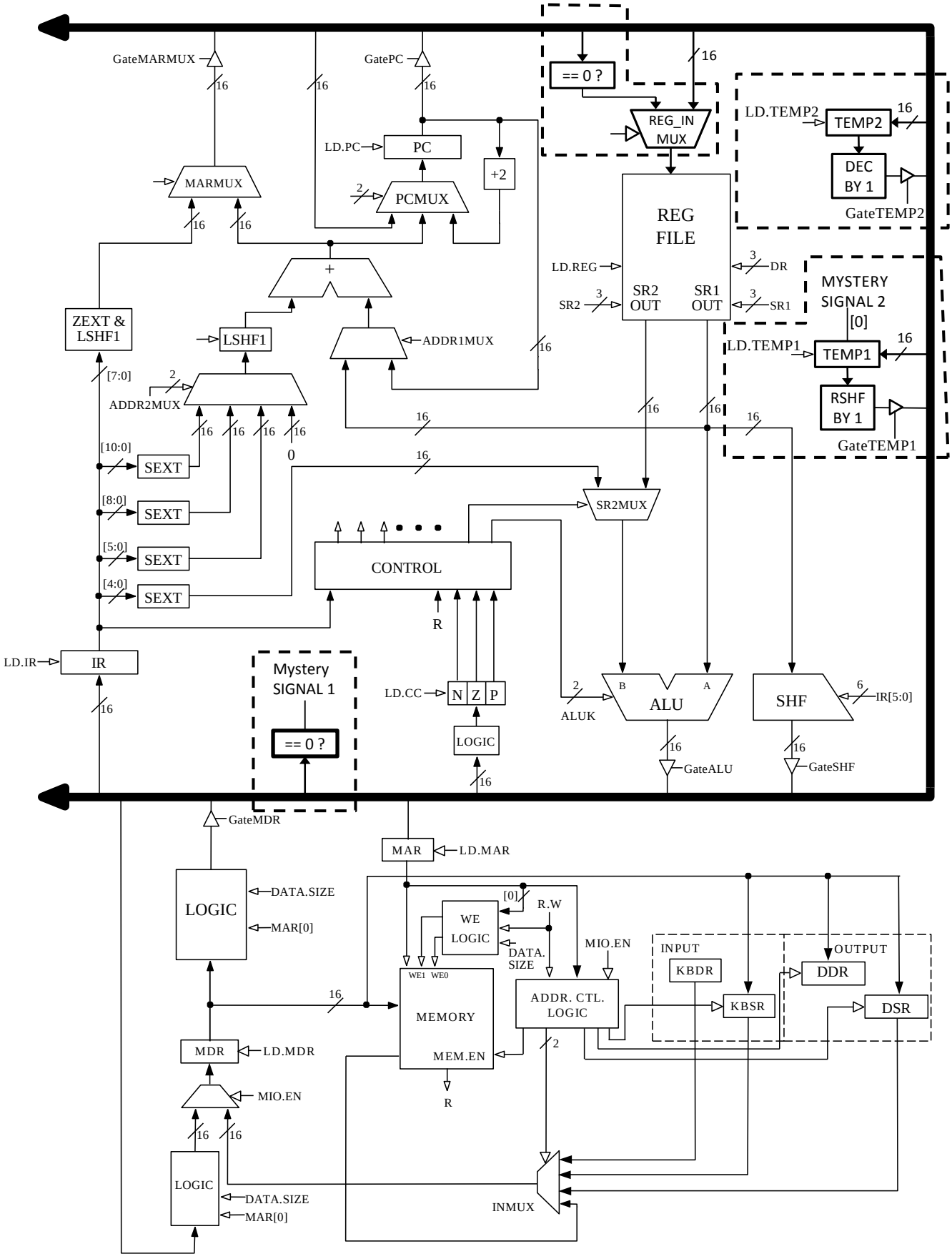
Describe what this instruction does.

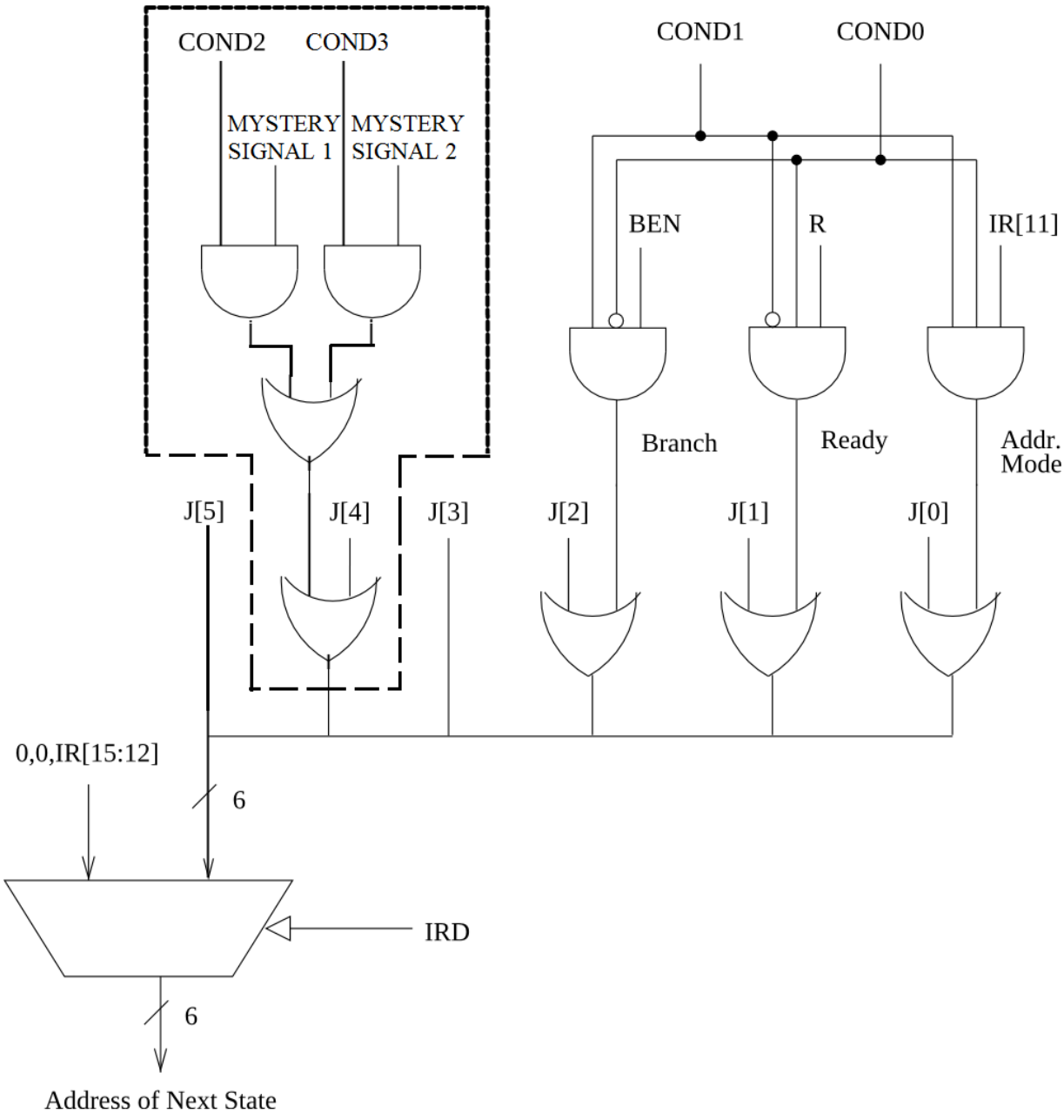
Determines if the 16-bit value stored in SR1 is a Palindrome itself.

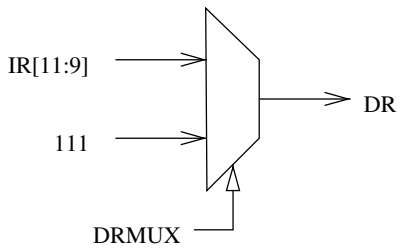
Code:

```
State 10: DR ← 0
State 11: TEMP1 ← value(SR1)
State 40: TEMP2 ← 16
State 37: DR = DR << 1
          if (TEMP1[0] == 0)
            goto State 45
          else
            goto State 61
State 61: DR = DR | 0x1
State 45: TEMP1 = TEMP1 >> 1
State 63: DEC TEMP2
          if (TEMP2 == 0)
            goto State 53
          else
            goto State 37
State 53: DR = DR ^ SR1
```

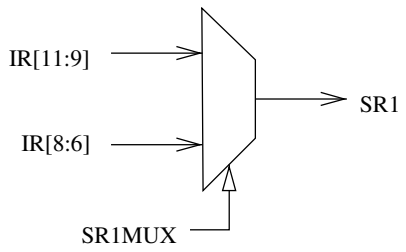
Initials:


GateMARMUX
GatePC
== 0 ?
16
REG_IN MUX
LD.TEMP2
TEMP2
DEC BY 1
GateTEMP2
LD.PC
PC
+2
MARMUX
PCMUX
REG FILE
LD.REG
DR
SR2
SR2 OUT
SR1 OUT
SR1
MYSTERY SIGNAL 2
[0]
LD.TEMP1
TEMP1
RSHF BY 1
GateTEMP1
ZEXT & LSHF1
LSHF1
ADDR1MUX
ADDR2MUX
[7:0]
[10:0]
[8:0]
[5:0]
[4:0]
SEXT
SR2MUX
CONTROL
R
LD.IR
IR
Mystery SIGNAL 1
LD.CC
N Z P
LOGIC
ALUK
ALU
SHF
IR[5:0]
GateALU
GateSHF
GateMDR
MAR
LD.MAR
DATA.SIZE
MAR[0]
WE LOGIC
R.W
MIO.EN
INPUT
OUTPUT
KBDR
DDR
KBSR
DSR
WE1 WE0
MEMORY
MEM.EN
ADDR. CTL. LOGIC
MDR
LD.MDR
INMUX

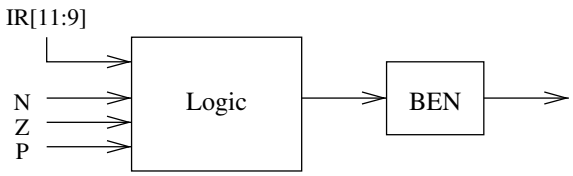

COND2
COND3
MYSTERY SIGNAL 1
MYSTERY SIGNAL 2
COND1
COND0
BEN
R
IR[11]
Branch
Ready
Addr. Mode
J[5]
J[4]
J[3]
J[2]
J[1]
J[0]
0,0,IR[15:12]
6
IRD
6
Address of Next State



(a)

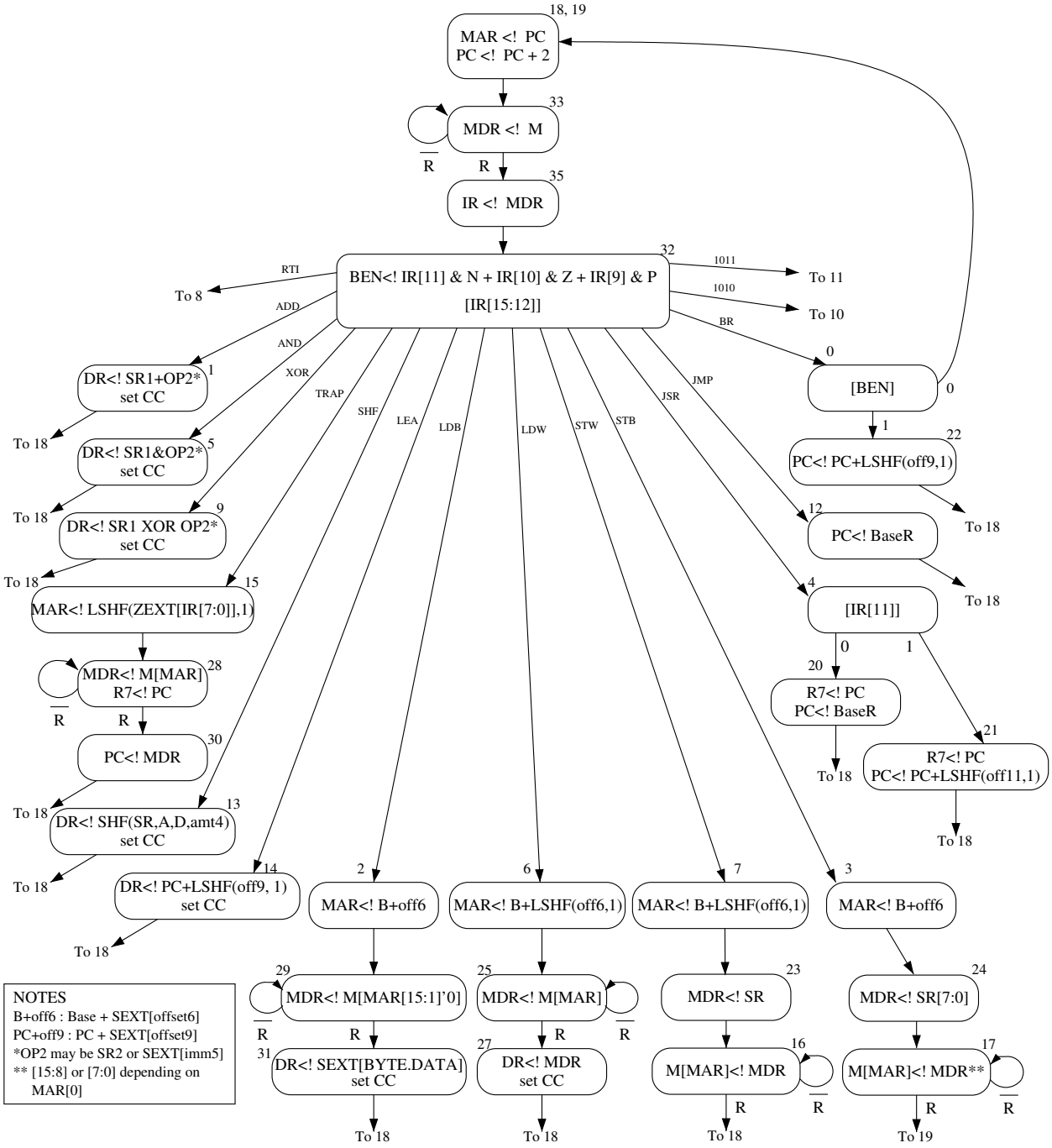


(b)



(c)

Initials:


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
ADD
AND
XOR
TRAP
SHF
LEA
LDB
LDW
STW
STB
JMP
JSR
1011
To 11
1010
To 10
BR
0
[BEN]
0
1
22
PC<! PC+LSHF(off9,1)
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
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
2
MAR<! B+off6
6
MAR<! B+LSHF(off6,1)
7
MAR<! B+LSHF(off6,1)
3
MAR<! B+off6
29
MDR<! M[MAR[15:1]'0]
R
R
31
DR<! SEXT[BYTE.DATA]
set CC
To 18
25
MDR<! M[MAR]
R
R
27
DR<! MDR
set CC
To 18
23
MDR<! SR
16
M[MAR]<! MDR
R
R
To 18
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]