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.

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])

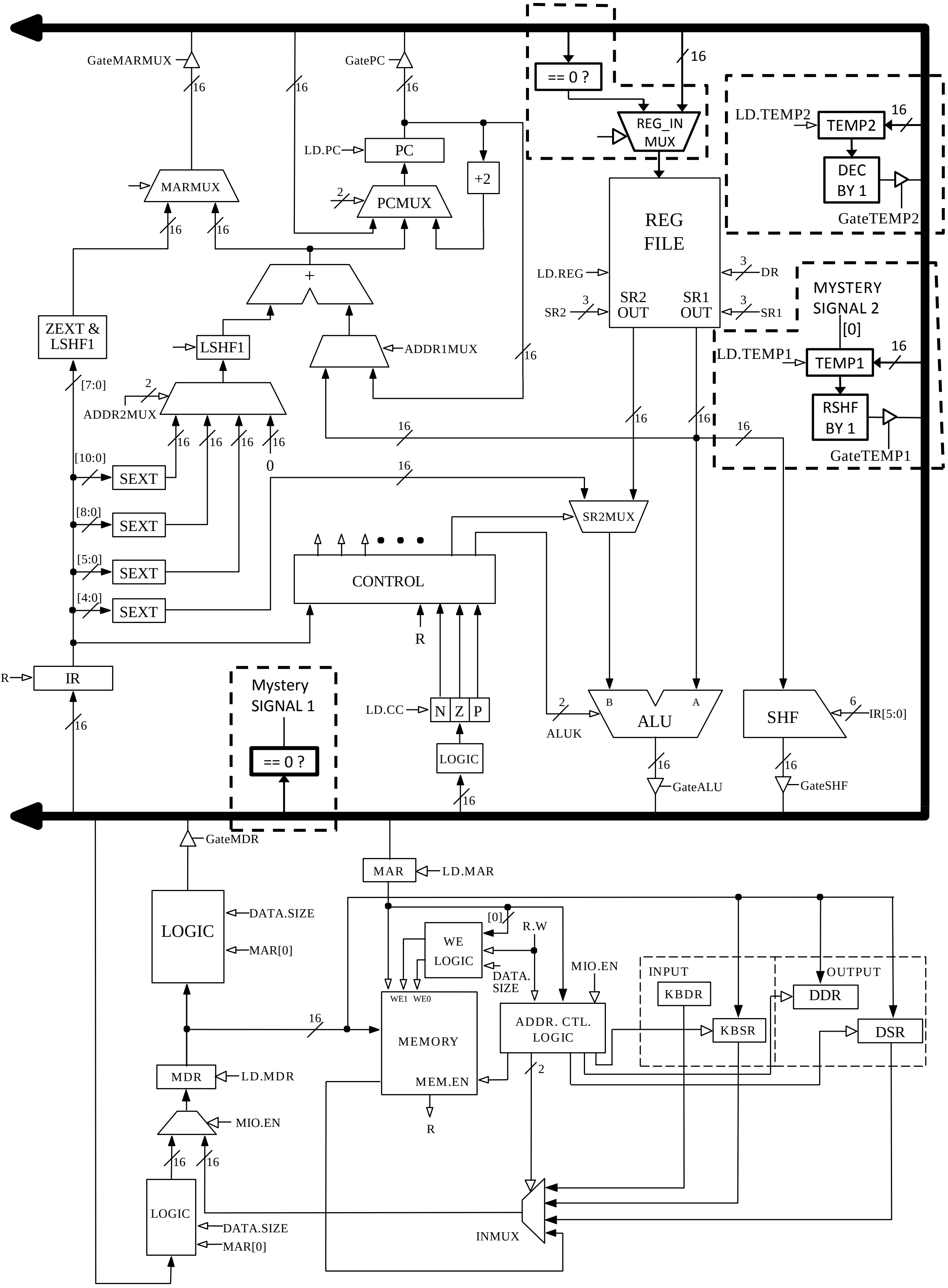
101101 (45)	COND_{0000}	111111	GateTEMP1, LD.TEMP1
111111 (63)	COND_{0100}	100101	GateTEMP2, LD.TEMP2
110101 (53)	COND_{0000}	010010	GateALU, ALUK = XOR, SR1MUX = DR (IR[11:9]) LD.REG, DRMUX = DR (IR[11:9]), Reg_IN_MUX = EQ0

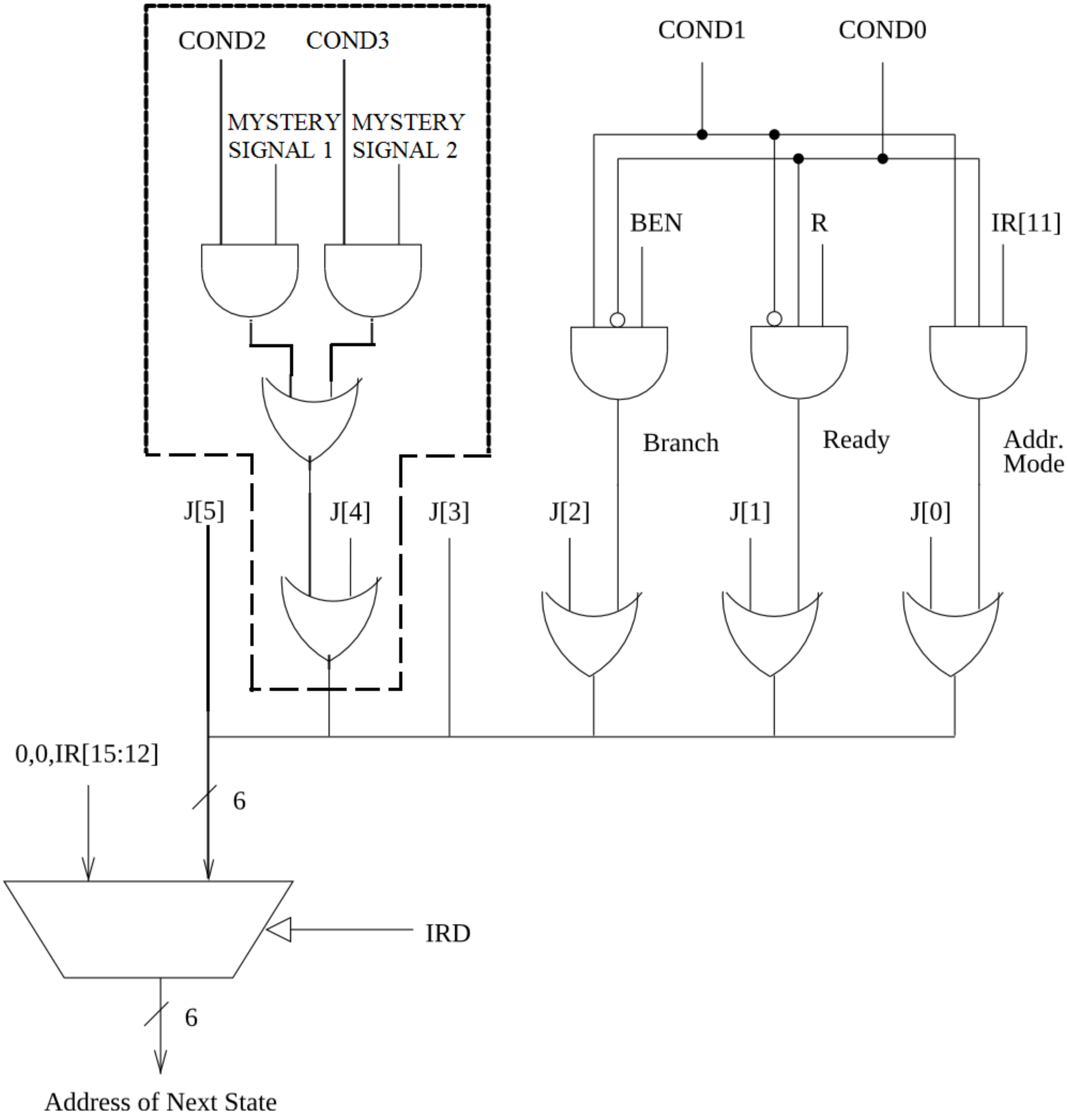
Describe what this instruction does.

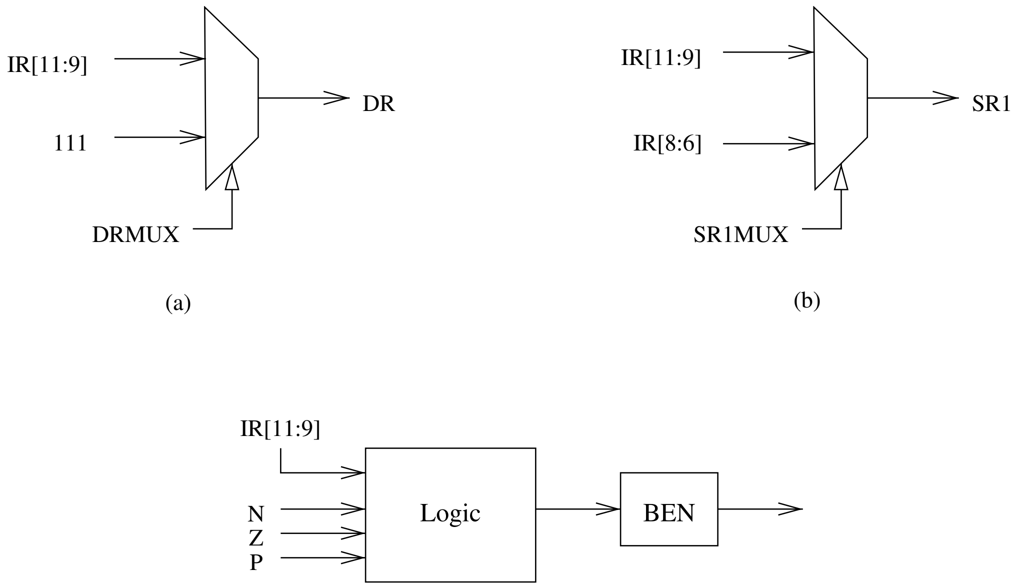
Determines if the 16-bit value stored in SR1 is a Palindrome itself.

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
```


GateMARMUX
16
16
GatePC
16
== 0 ?
16
REG_IN
MUX
LD.TEMP2
16
TEMP2
DEC
BY 1
GateTEMP2
LD.PC
PC
+2
MARMUX
16
16
2
PCMUX
REG
FILE
LD.REG
3
DR
SR2
OUT
SR1
OUT
3
SR2
3
SR1
MYSTERY
SIGNAL 2
[0]
LD.TEMP1
16
TEMP1
RSHF
BY 1
GateTEMP1
+
ZEXT &
LSHF1
LSHF1
ADDR1MUX
16
[7:0]
2
ADDR2MUX
16
16
16
16
0
16
16
16
16
[10:0]
SEXT
[8:0]
SEXT
[5:0]
SEXT
[4:0]
SEXT
SR2MUX
CONTROL
R
LD.IR
IR
16
Mystery
SIGNAL 1
== 0 ?
LD.CC
N
Z
P
LOGIC
16
2
ALUK
B
A
ALU
16
GateALU
SHF
6
IR[5:0]
16
GateSHF
GateMDR
MAR
LD.MAR
LOGIC
DATA.SIZE
MAR[0]
[0]
R.W
WE
LOGIC
DATA.
SIZE
MIO.EN
INPUT
KBDR
OUTPUT
DDR
WE1
WE0
16
MEMORY
ADDR. CTL.
LOGIC
KBSR
DSR
MEM.EN
2
MDR
LD.MDR
R
MIO.EN
16
16
LOGIC
DATA.SIZE
MAR[0]
INMUX

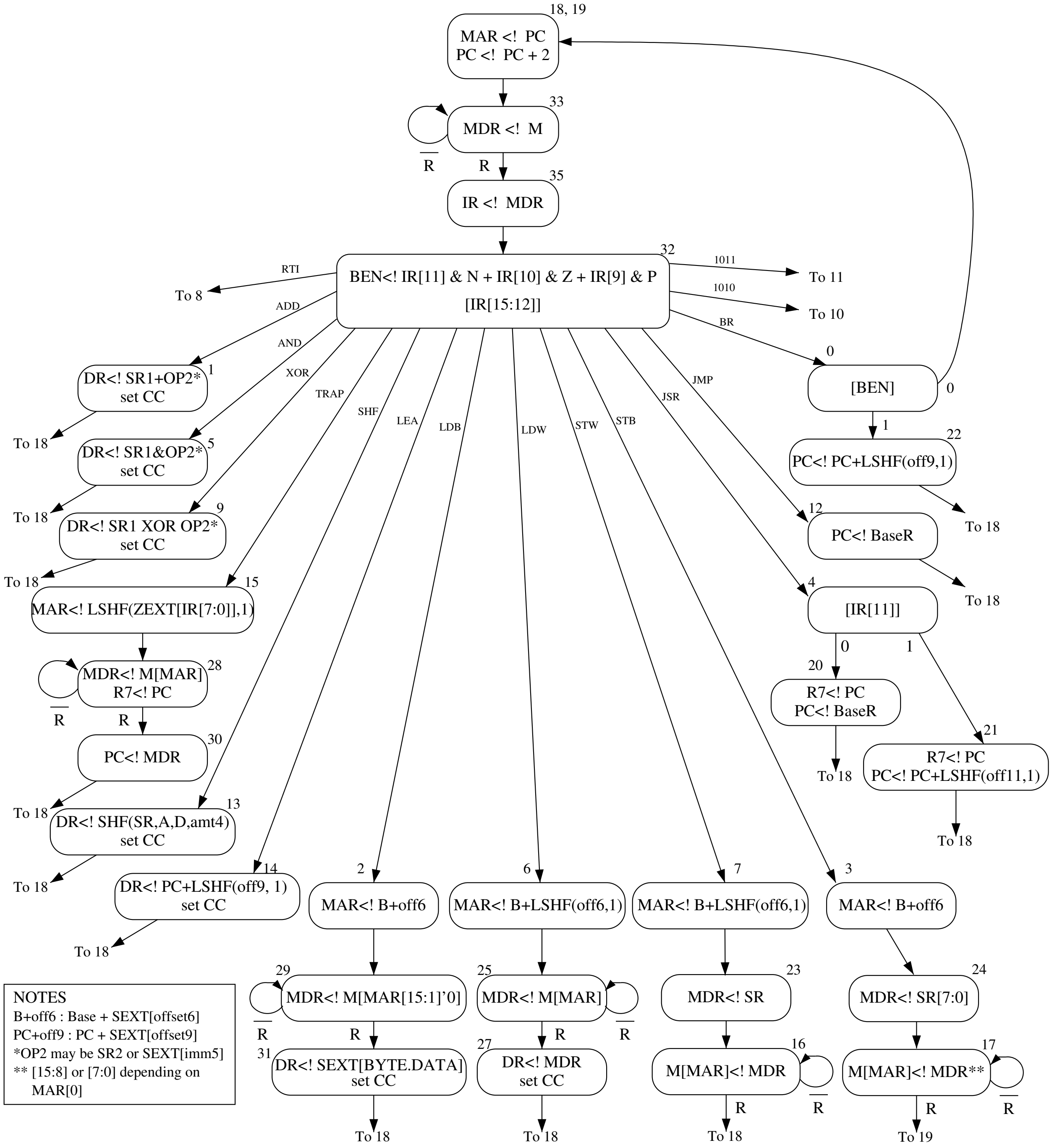

COND2
COND3
MYSTERY SIGNAL 1
MYSTERY SIGNAL 2
COND1
COND0
BEN
R
IR[11]
Branch
Ready
Addr. Mode
J[5]
J[4]
J[3]
J[2]
J[1]
J[0]
0,0,IR[15:12]
6
IRD
6
Address of Next State



(a)

(b)

(c)



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]