State Diagram:

Interrupt/Exception:

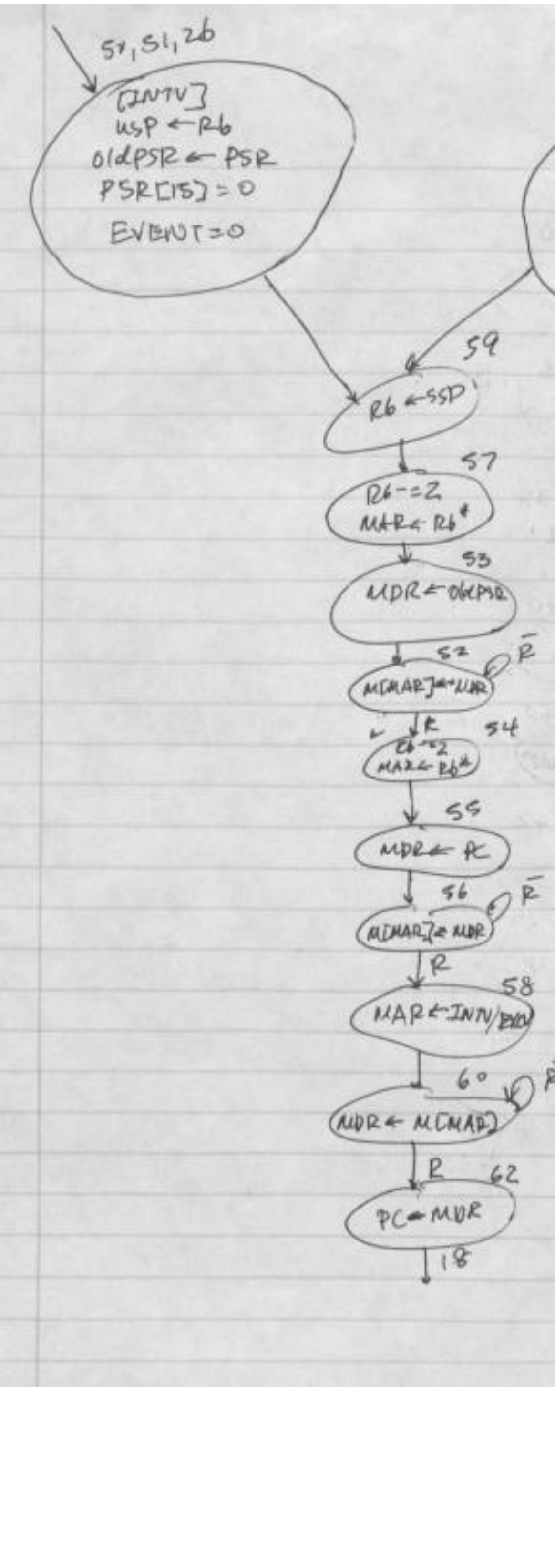
26,50,51: Load INTV with the correct value. Load the User Stack Pointer (USP) with value in R6. Load the oldPSR register with PSR. Sets PSR to Supervisor Mode. Set EVENT = 0 to indicate interrupt. Clears the interrupt flag.

63: Same as 26, 50, 51 except EXCV is loaded, and EVENT is set to 1 to indicate exception.

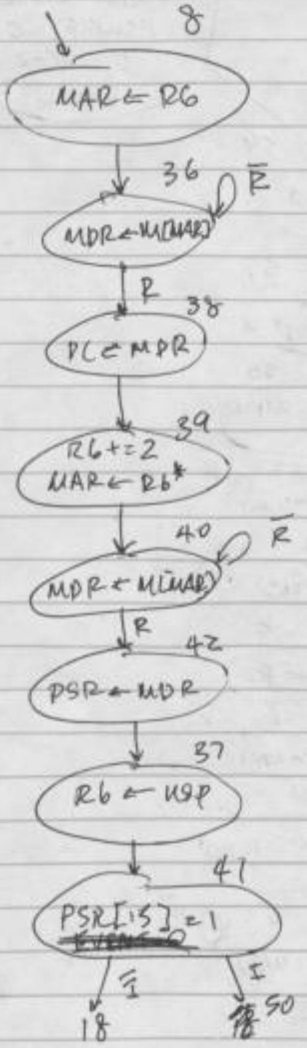
59: Loads R6 with Supervisor Stack Pointer (SSP)
57: Subtracts R6 by 2 and loads into MAR
53: Loads MDR with oldPSR
52: Stores oldPSR into address defined by R6
54: Subtracts R6 by 2 and loads into MAR
55: Loads MDR with PC
56: Stores PC into address defined by R6
58: Loads MAR with INTV or EXCV depending on EVENT bit
60: Load value stored at that location
62: Load MDR into PC

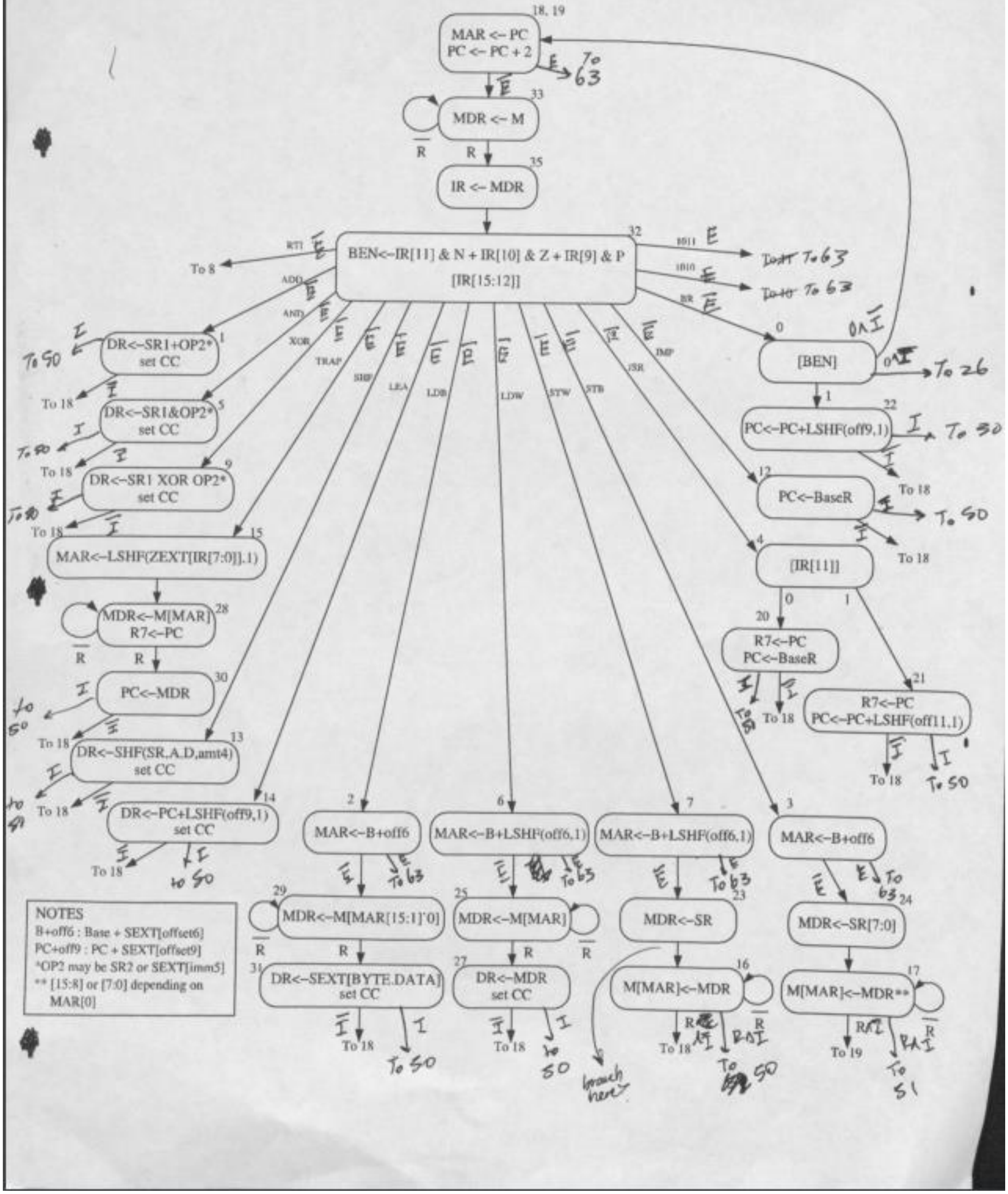
RTI:

8: Load MAR with R6
36: Load value defined by R6
38: Load value from MDR into PC
39: Add R6 by 2 and load into MAR
40: Load value defined by R6
42: Load PSR with value from MDR
37: Load R6 with User Stack Pointer (USP)
41: Switch back to User Mode


57, 51, 26
[INTV]
USP ← R6
oldPSR ← PSR
PSR[15] = 0
EVENT = 0
63
[EXCV]
USP ← R6
oldPSR ← PSR
PSR[15] = 0
PC = PC-2
EVENT = 1
59
R6 ← SSP
57
R6 -= 2
MAR ← R6*
53
MDR ← oldPSR
52
$\bar{R}$
M[MAR] ← MDR
R
54
R6 -= 2
MAR ← R6*
55
MDR ← PC
56
$\bar{R}$
M[MAR] ← MDR
R
58
MAR ← INTV/EXCV
60
$\bar{R}$
MDR ← M[MAR]
R
62
PC ← MDR
18

RTI


8
MAR ← R6
36
R̄
MDR ← M[MAR]
R
38
PC ← MDR
39
R6+=2
MAR ← R6*
40
R̄
MDR ← M[MAR]
R
42
PSR ← MDR
37
R6 ← USP
41
PSR[15] = 1
Ī
I
18
50



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]

Datapath:

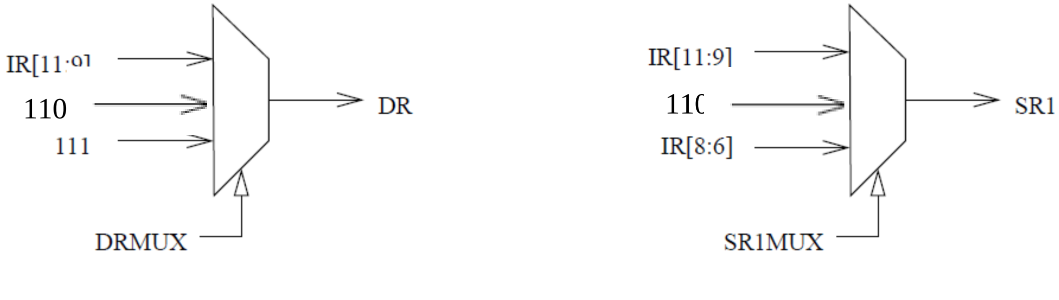
The GateStack structure is used to implement stack switching. It can output changes to the R6 register by passing in a +/- 2 to the SR1out. It’s also used to pass either the USP or SSP to the bus as needed, and only the USP ever needs to be loaded.

The additional -2 adder option into PCMUX is to handle exceptions.

The MARMUX2 structure is used for altering values from the vector table, before storing into the MAR.

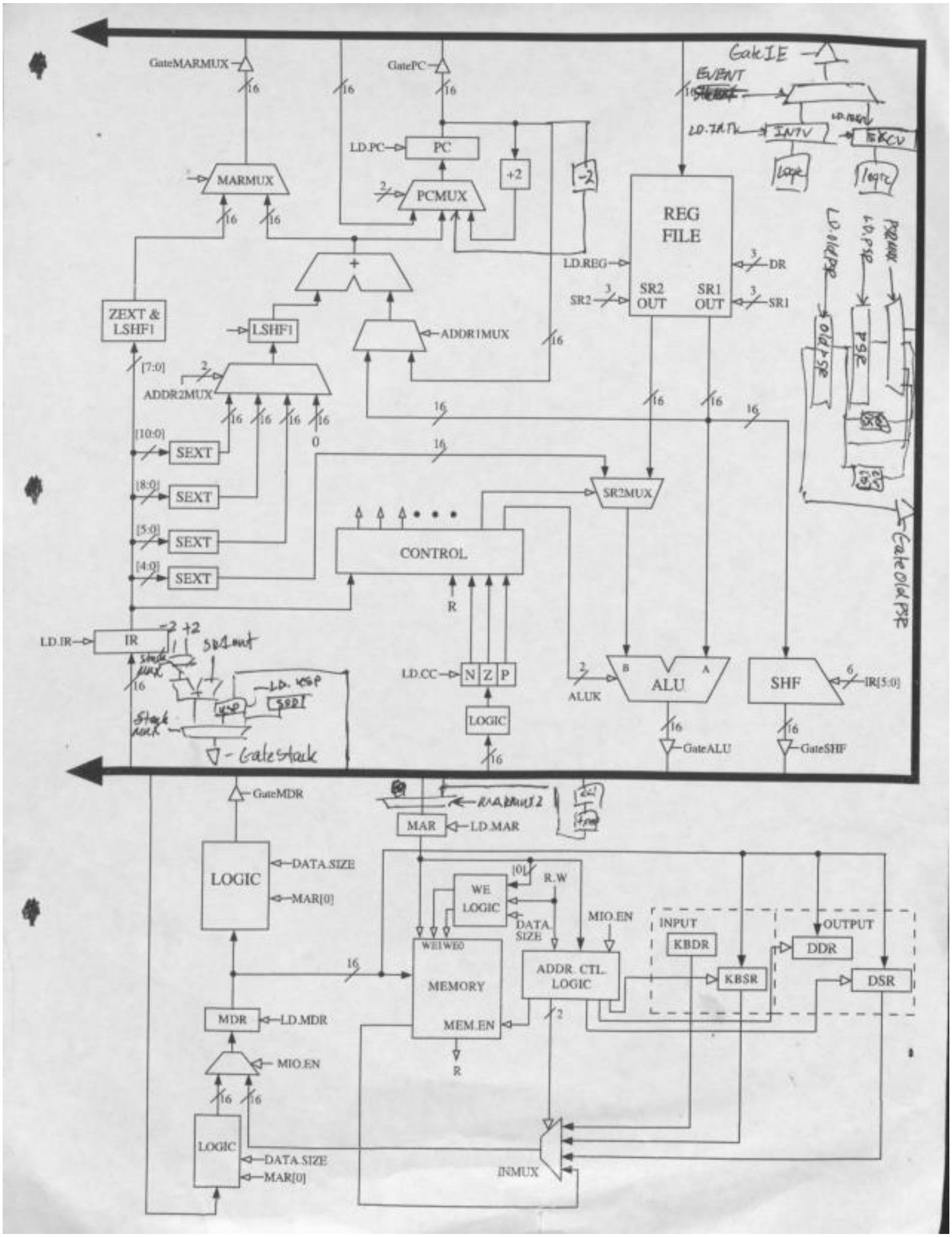
The PSR structure is used for efficient saving of the old PSR before saving and to effectively either clear bit 15, set bit 15, or obtain data from the BUS.

The GateIE structure is used to pass appropriate EXCV or INTV values depending on the EVENT that has occurred. These registers are loaded by additional logic, which will be set by analyzing the current state of the machine.



(a)

(b)


GateMARMUX
GatePC
16
MARMUX
LD.PC
PC
+2
-2
PCMUX
2
+
LSHF1
ZEXT &
LSHF1
[7:0]
ADDR1MUX
ADDR2MUX
[10:0]
SEXT
[8:0]
SEXT
[5:0]
SEXT
[4:0]
SEXT
0
REG
FILE
LD.REG
DR
SR2
SR2
OUT
SR1
OUT
SR1
3
GateIE
EVENT
LD.INTV
INTV
LD.EXCV
EXCV
logic
logic
LD.oldPSR
LD.PSR
PSRMUX
old PSR
PSR
GateOldPSR
SR2MUX
CONTROL
R
LD.CC
N
Z
P
LOGIC
ALU
ALUK
B
A
SHF
IR[5:0]
6
GateALU
GateSHF
LD.IR
IR
-2
+2
SR1out
stack
mux
LD.USP
USP
SSP
Stack
mux
Gatestack
GateMDR
MARMUX2
MAR
LD.MAR
LOGIC
DATA.SIZE
MAR[0]
[0]
R.W
WE
LOGIC
DATA.
SIZE
MIO.EN
WE1WE0
MEMORY
MEM.EN
R
ADDR. CTL.
LOGIC
INPUT
KBDR
KBSR
OUTPUT
DDR
DSR
MDR
LD.MDR
MIO.EN
LOGIC
DATA.SIZE
MAR[0]
INMUX

New Control Signals:

LD.EXCV, LD.PSR, LD.OLDPSR, LD.EVENT, and LD.USP are used to control when these registers are loaded.

INTFLAGCLR is used to clear the interrupt flag after entering states 50,51, or 26 to indicate that the interrupt is being handled.

CCUPDATE is used to update condition codes during the RTI instruction when PSR is loaded from the supervisor stack.

PCMUX has an additional possibility (3) to indicate PC = PC-2.

DRMUX can now have the destination register be set to R6 with value 2.

SR1MUX can now have the source register be set to R6 with value 2.

MARMUX2 is used to select what should be loaded into MAR

0: BUS
1: BUS << 1 + x200

PSRMUX is used to select which value will be loaded into the PSR.

0: update CC portion of PSR
1: clear PSR[15]
2: set PSR[15]
3: BUS

STACKADDRMUX is used to select which value to add to SR1, which should be R6

0: -2
1: +2

STACKMUX is used to select which value should be sent to the bus for stack switching

0: STACKADDR
1: USP
2: SSP

EVENTMUX is used to select what should be loaded into the EVENT register.

Microsequencer:

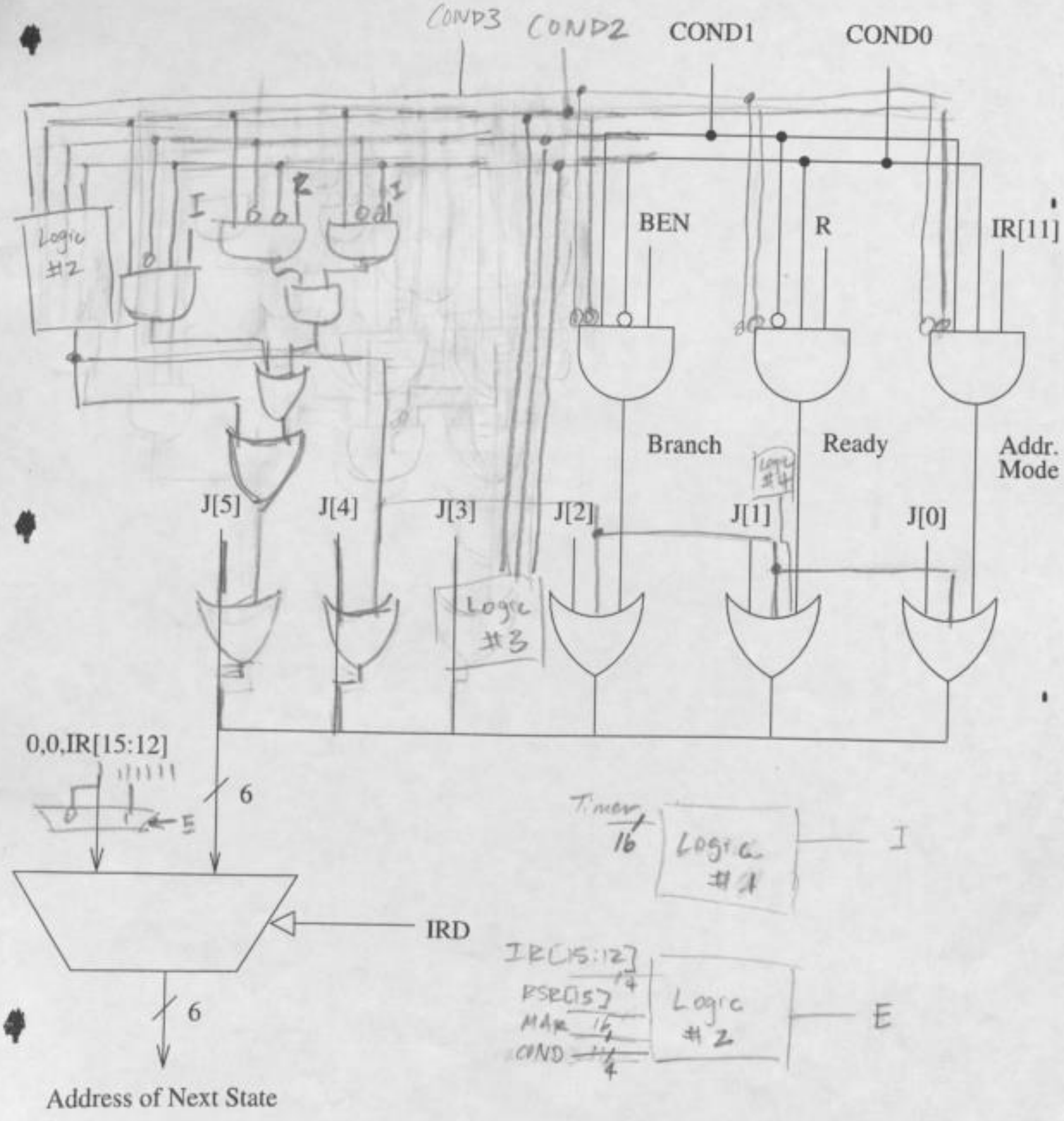
This was designed to simply exceptions, by sending all exceptions to the same state 63 (11111). This required an additional COND bit. There were specific combinations of condition bits meant to ensure which exceptions could occur where.

0110: unknown opcode
0111: protected or unaligned
1000: unaligned only

Other condition bits were used for interrupts.

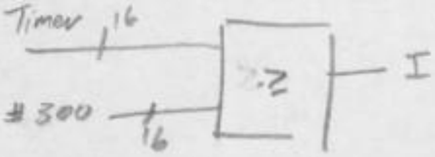
0101: normal check of interrupt flag
0100: check for ready bit and interrupt flag for memory accesses

Additional logic is used for the BEN state 0 because it should only branch to state 26 if the branch isn’t taken. An additional MUX is also added in case IRD = 1 to handle unknown opcode exceptions.

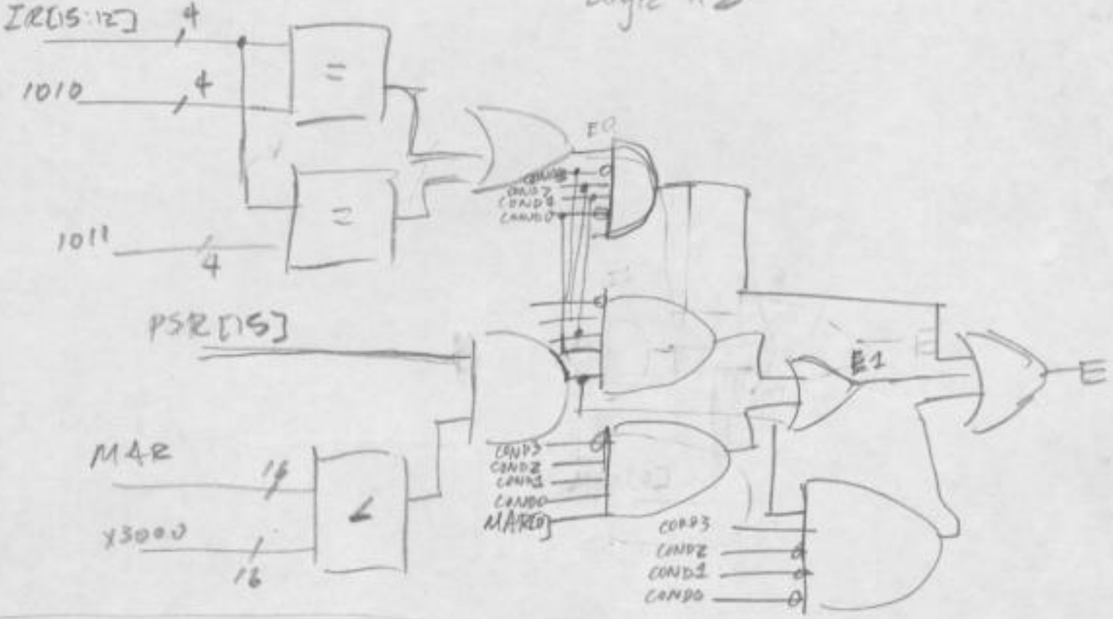


Logic on next page →

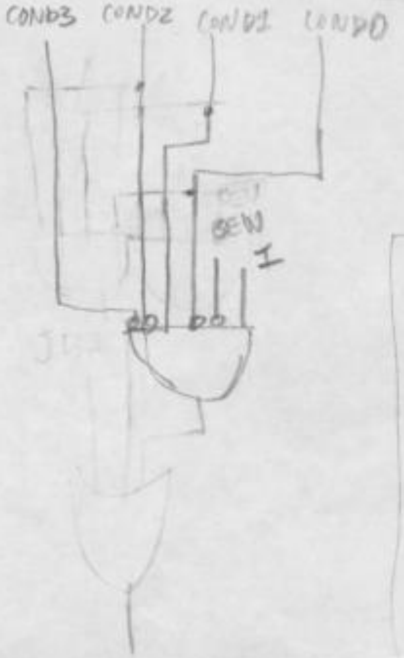
Logic #1



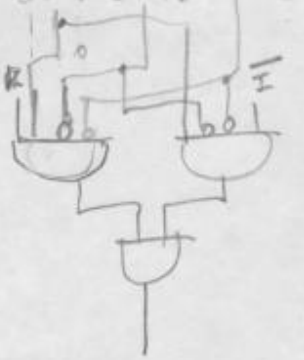
Logic #2



Logic #3



Logic #4



COND2
COND1
COND0
R
I