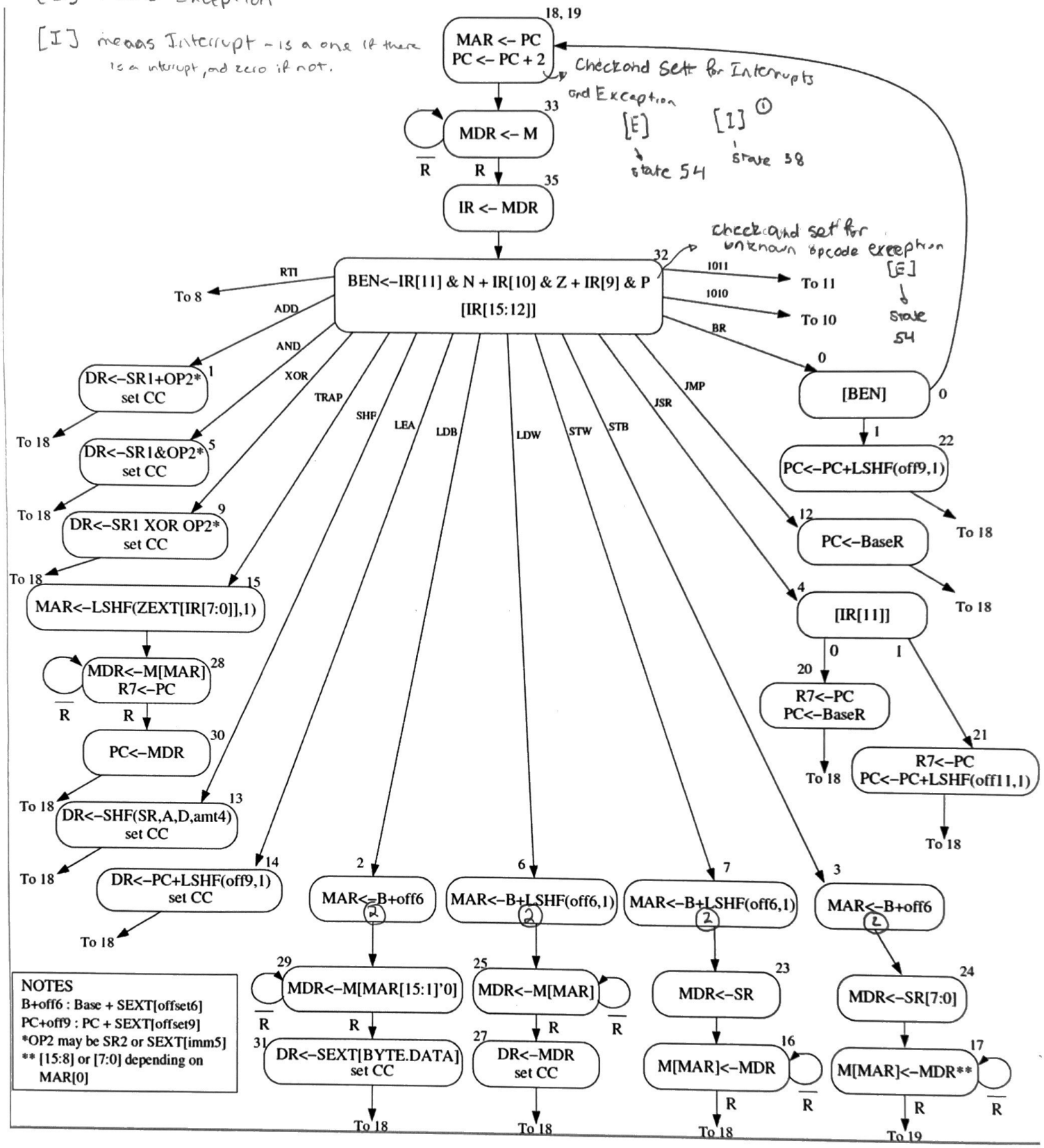
① State Path handles exceptions before interrupts - Implemented in microcontroller - happens in State 18

② At the end of the state, exceptions are checked and set. If [E] is one go to state 54.

[E] means Exception - is a one if there is an exception, and zero if not.

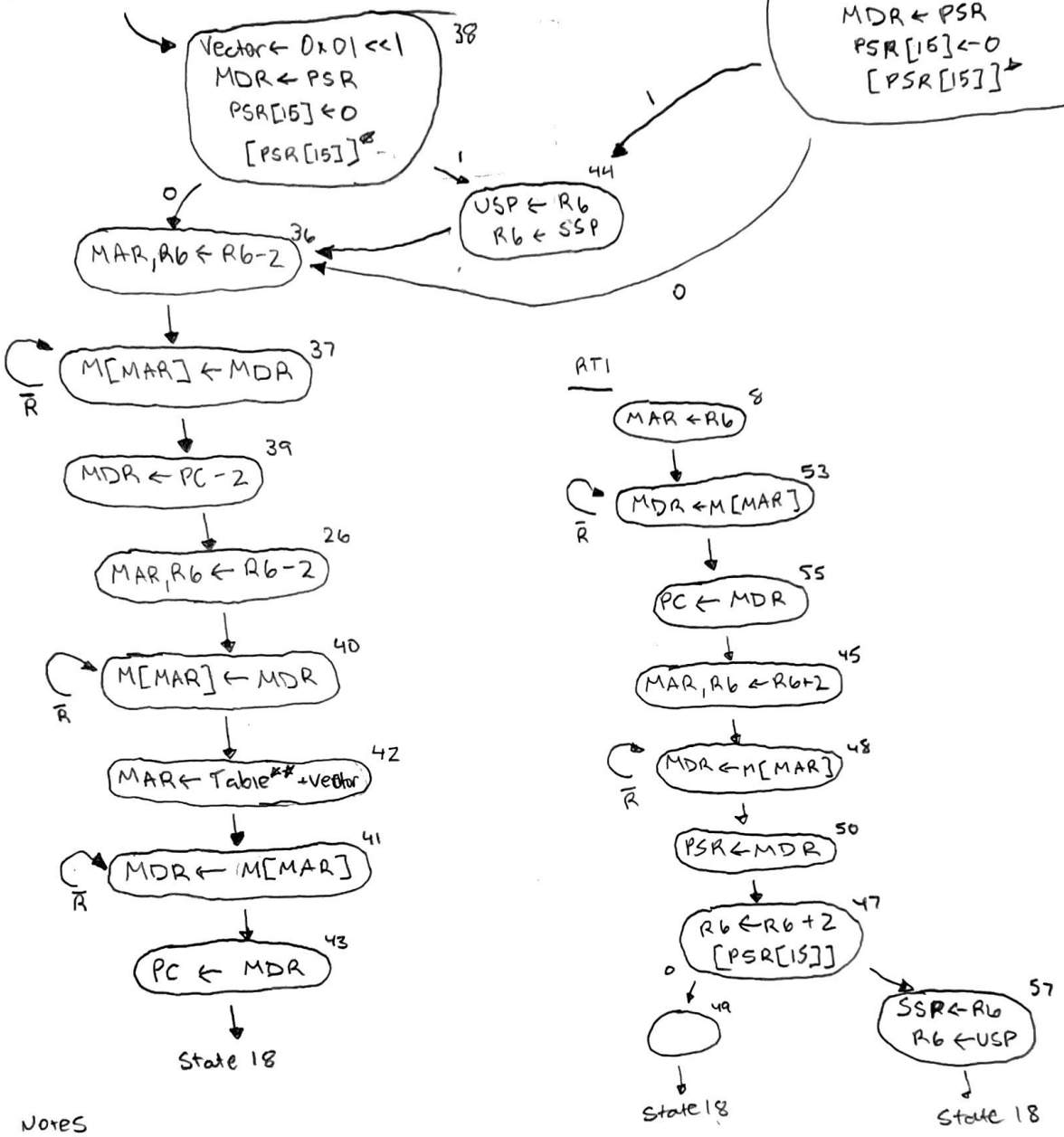
[I] means Interrupt - is a one if there is a interrupt, and zero if not.



Lab 4

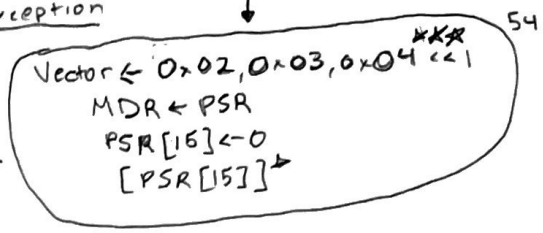
Interrupts

from State 18



Exception

from State 18, 19, 32, 2, 6, 7, or 3



RTI

Notes

* This is the old value of PSR[15] before PSR[15] gets changed to 0. PSR[15] also refers to the bit at position 15 for the register PSR
** Table is a global constant variable that is set to 0x0200. Acted as a constant value in my datapath.
*** Based on the exception that is called, different values are stored in the register vector. For protection exception the value of 0x02<<1 is stored in vector, 0x03<<1 for unaligned access exception, and 0x04<<1 for unknown opcode exception. "<<" means right shift by one.

Lab 5 states

Interrupts/Exception state

From state 18, 19, 15, 2, 6, 7, 3, 36, 26, 42, 8, 45 (RTI)

New states

States before are from Lab 4 and before

34: MAR ← PTBR[15:8] ' LSHF[VA[15:9], 1]
in[15:8] in[7:0]
→ no control signal
Saves MDR in register before

58: MDR ← M[MAR] → R̄

58: PTE ← MDR → check and set [E] bit /IRD = 2
(doesn't check for unaligned - checks for that in any other state)
checks for protection and page fault exception

46: MDR, PTE ← PTE | LSHF(RW, 1) + 1

60: M[MAR] ← MDR → R̄

62: MAR ← PTE[15:9] ' VA[8:0]
IRD = 3
→ control signal loads MDR in MDR register clk

return to state given by BS
(33, 33, 28, 29, 25, 23, 24, 37, 40, 41, 53, 48) respectfully

New additions to the state diagram from lab 4

State 18, 19, 15, 2, 6, 7, 3 - at the end of the state save the next state (J value) into the register BS. Also in these states only check only for unaligned access exception instead of all.

State 33 - check for unknown opcode exception only - was doing this before but just stating where.

State 36, 26, 42, 8, 45 - In these states check only for unaligned access exception instead of all exception's since the MAR has the value of VA.

In all these states (18, 19, 15, 2, 6, 7, 3, 36, 24, 42, 8, 45) - The value of MAR is stored into the register VA.

State Diagram

New states

State 34 – Coming from multiple other states, state 34 first saves the value of MDR before doing any operations in the TempMDR register. This is controlled by the LD. protectMDR which loads the value of the MDR into the TempMDR (they are connected by a wire). The state then loads the value of the address of the PTE of the page containing the VA in the MAR using the bus. The value of PTBR [15:8] (stored in bits 15-8) is added with LSHF (VA [15:9],1) (stored in bits 7-0) and that value is put on the bus using the gateADDERVA which is then loaded into the MAR.

State 56 – This state comes from state 34 unconditionally. The MDR will be getting the value at address MAR through the five cycles. This is a memory read state where the value at address MAR gets stored in MDR. In this state – the value is the PTE

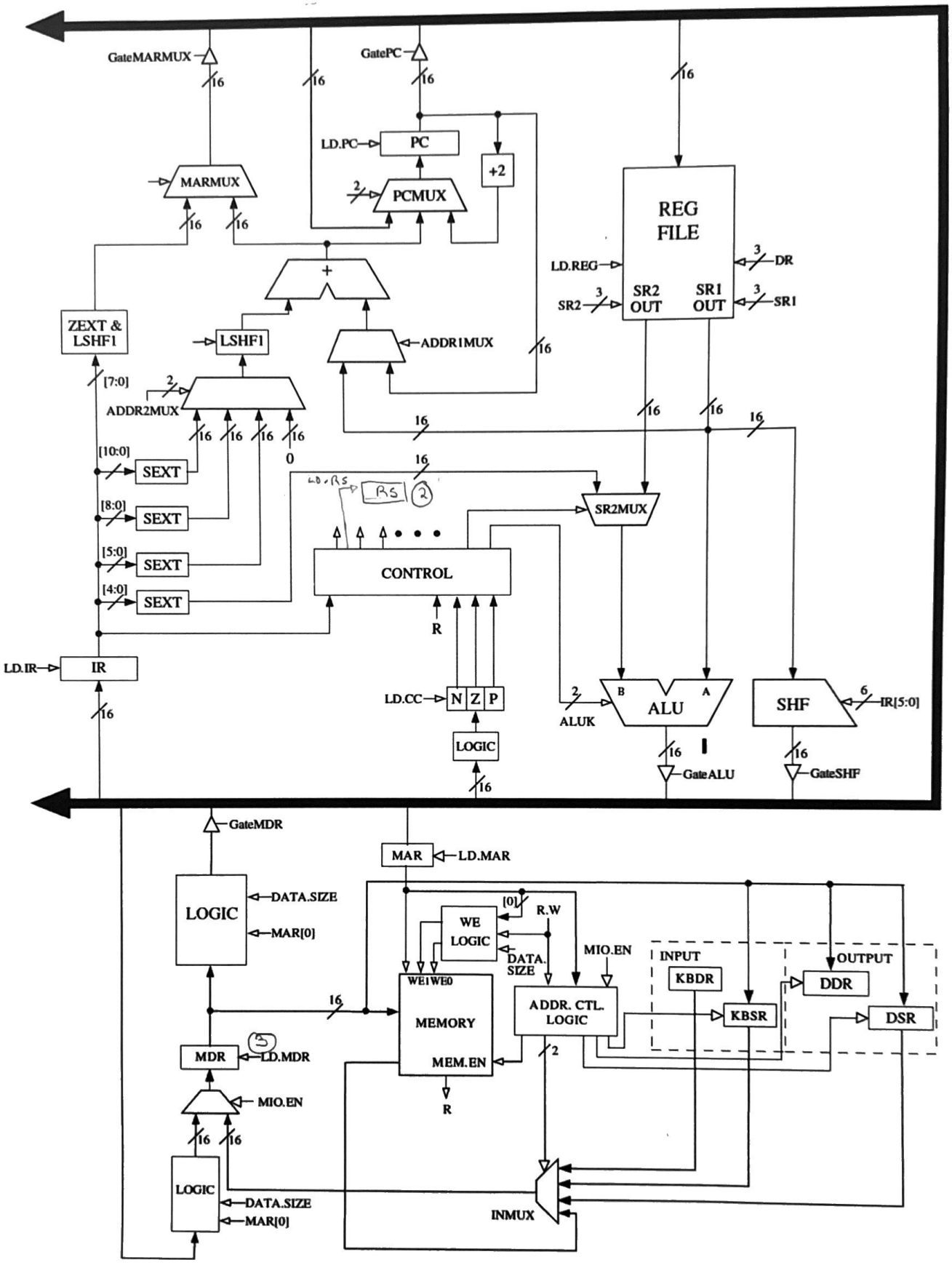
State 58 – This state comes from state 56 unconditionally. In this state the value of MDR is put on the bus using the gateMDR. The value on the bus is then loaded in the PTE through the control signals of LD.PTE and the PTEMux have the value of 0 in order to load the value of the bus into the PTE. This state also checks for protection and page fault exception using the value in the MDR/bus value. This is the value of the PTE and contains values needed for checking for protection and page fault exception. If there are exceptions then the state goes to state 54, if not it goes 46

State 46 – This state comes from state 58. The value of PTE | LSHF(NextLatches.RW,1) +1 is put on the bus using the GatePTE. This state changes the PTE 's modified bit if the memory is being written to and it changes the PTE's reference bit all the time in this state. The value from the bus is then stored in both the PTE and MDR.

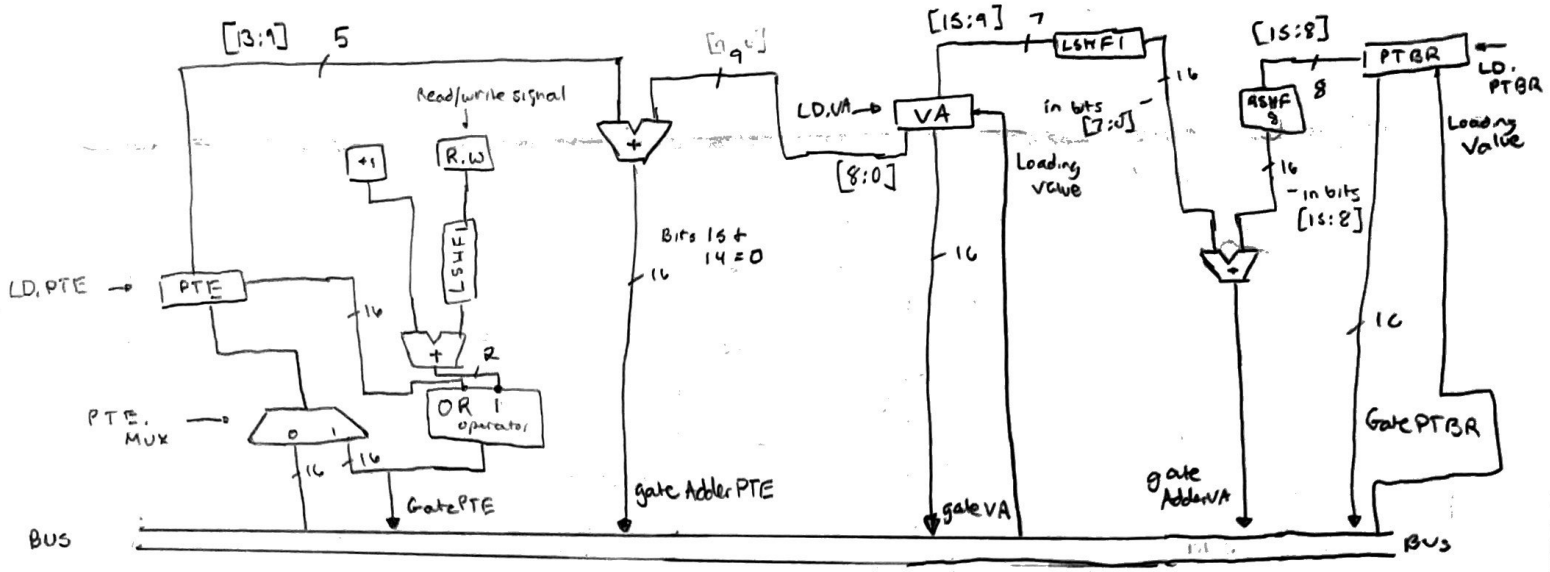
State 60 – This state unconditionally comes from state 46. In this state MDR is getting stored at address MAR. This state takes 5 cycles and is a memory write state

State 62 – This state unconditionally comes from state 60. In this state 00'PTE [13:9]'VA [8:0] is calculated by an adder and put on the bus using GateADDERPTE. This value on the bus is then loaded into the Mar using LD.MAR. This is the physical address that is found using the PTE and VA. The state then goes to the state stored in RS which was loaded in the state before state 34. For instance, if state 18 doesn't have any interrupts or exceptions it saves state 33 in RS and goes to state 34.

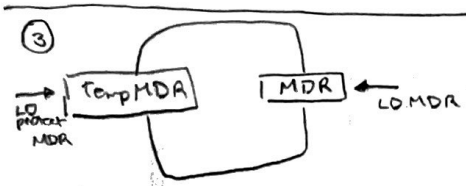
I used 59 states in total. This includes state 10 and 11 even though I never utilized them.


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

①


[3:9]
5
Read/write signal
+1
R.W
LSHF1
LD.PTE
PTE
16
+
R
PTE. MUX
0
1
OR 1 operator
16
16
GatePTE
BUS
[9:0]
Bits 15 & 14 = 0
16
gate AdderPTE
[15:9]
7
LSHF1
LD.VA
VA
[8:0]
16
Loading value
16
gate VA
in bits [7:0]
16
[15:8]
PTBR
LD. PTBR
LSHF 8
8
16
- in bits [15:8]
Loading Value
16
Gate PTBR
gate AdderVA
BUS

③


LD product MDR
tempMDA
MDR
LD MDR

Datapath and control signals
Labeled all the structures on the datapath

Control signals – LD.PTE PTEMUX, GATEPTE, GateADDERPTE, LD. VA, gateVA, gateADDERVA, LD.PTBR, GATEPTBR. Has the registers PTE, VA, PTBR. Next structure has LD.RS as a control signal and register RS. Next structure has TEMPMDR as a register and LD. protectMDR as a control signal – connected to MDR.

The first structure that I added to the Data path can be broken up into pieces. The left side of the structure which includes the PTE register, LD. PTE, PTEMUX and the adder with the OR operator that has an output to the bus through the GatePTE is used for changing the value of the PTE's reference bit and modified bit accordingly. The state that uses this structure is state 46 which changes the PTE based on what operation is happening in the frame.

The control signals added for the first part of the first structure are LD.PTE, PTEMUX, GatePTE. The GatePTE outputs the value of PTE | LSHF(NextLatches.RW,1) +1 which is responsible for outputting the value of the modified PTE on the bus. The PTEMUX is responsible for choosing the correct value you want in the PTE register. If the PTEMUX has the value of 0 then it chooses the value from the bus to output to the PTE register. If the PTEMUX has the value of 1 then it chooses the value of PTE | LSHF(NextLatches.RW,1) +1 to output to the register PTE. Lastly the LD.PTE is a control signal that loads the value that is inputted to the register.

The second part of the first structure includes GateADDERPTE, gateVA, register VA, LD. VA. This part of the structure is responsible for putting the physical address in the MAR. This is part of the last step in VA translation as seen in state 62.

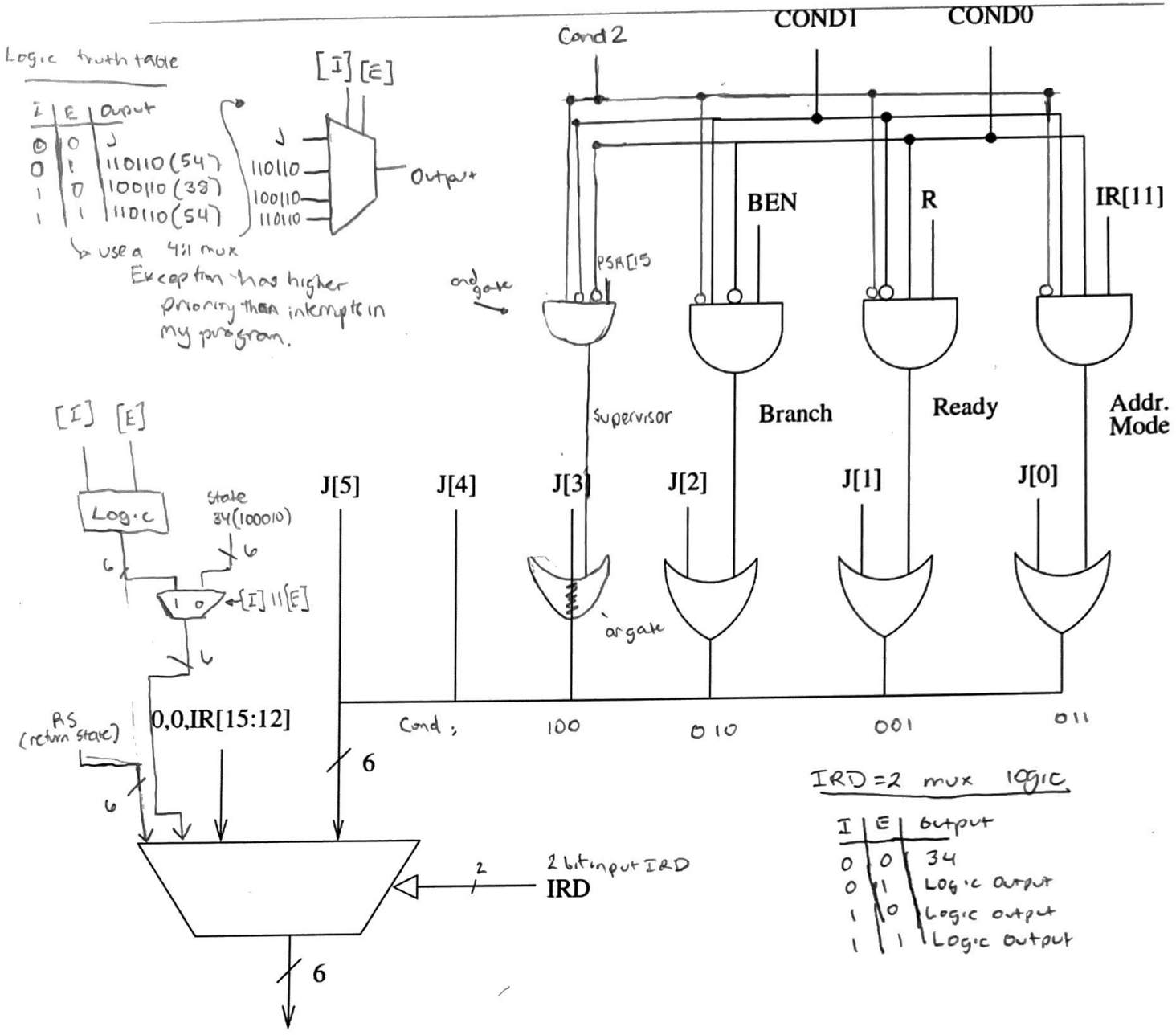
The control signals that are part of the second part of the first structure are LD.VA, gateVA, and GateADDERPTE. The GateADDERPTE is responsible for outputting the adder value to the bus which has the value of 00'PTE [13:9]'VA [8:0]. The gateVA is responsible for outputting the value of VA into the bus. Lastly the LD.VA is responsible for loading the value from the bus into the register VA.

The third part of the first structure includes gateADDERVA, LD. PTBR, gatePTBR, register PTBR. This part of the structure is responsible for putting the value of the address of the PTE of the page containing the VA. One state that uses this part of the structure is state 34 which loads the value of PTBR [15:8] (stored in bits 15-8) + LSHF (VA [15:9],1) (stored in bits 7-0) into the MAR.

The control signals that are part of the third part of the first structure are LD. PTBR, gatePTBR, and gateADDERVA. The LD. PTBR is responsible for loading the value from the bus into register PTBR whenever the control signal is one. The gatePTBR is responsible for outputting the value of the PTBR onto the register. The gateADDERVA is responsible for outputting the value PTBR [15:8] (stored in bits 15-8) + LSHF (VA [15:9],1) (stored in bits 7-0) to the bus using an adder.

The second structure which includes a RS (return state) register and a LD.RS is responsible for saving the next state after the program goes into the virtual translation path. For instance, the state 2 would have a control signal of 1 for LD. RS and save the value of 29 into the RS register.

The third structure includes a TempMDR register, and a control signal LD. protectMDR. This structure is responsible for saving the MDR during state 34 and by having the LD. protectMDR signal one and then having the LD.MDR signal one in state 62 to load the value in TempMDR into MDR. This structure is important because in the VA translation states you are changing the value of MDR which can lead to problems if you have an important value in the MDR.


Logic truth table
I | E | ouput
0 | 0 | J
0 | 1 | 110110 (54)
1 | 0 | 100110 (38)
1 | 1 | 110110 (54)
use a 4:1 mux
Exception has higher priority than interrupts in my program.
[I] [E]
J
110110
100110
110110
Output
Cond 2
COND1
COND0
BEN
R
IR[11]
PSR[15]
and gate
Supervisor
Branch
Ready
Addr. Mode
J[5]
J[4]
J[3]
J[2]
J[1]
J[0]
or gate
Log.c
State 34(100010)
6
1 0
[I] || [E]
PS (return state)
0,0,IR[15:12]
Cond:
100
010
001
011
2
2 bit input IRD
IRD
Address of Next State
IRD=2 mux logic
I | E | output
0 | 0 | 34
0 | 1 | Logic output
1 | 0 | Logic output
1 | 1 | Logic output

Describe

Microsequencer

The major changes that I made to the microsequencer was adding the mux for IRD=2 and adding the IRD=3 value in the mux.

The mux for IRD=2 is important to have as it chooses whether the next state should go to VA translation or interrupt or exception. If there is an interrupt or exception the next state is chosen by the logic block, but if there is no interrupt or exception then the next state is 34 which is the next state. For instance, in state 18, if there is an exception, state 18 goes to 38 but if there is no exception or interrupt, state 18 goes to state 34.

The IRD=3 value is for returning back to the next state that called the VA translation. For instance, if there are no interrupts or exceptions, before state 2 goes to state 34, the value of 29 is stored in the RS register since that's the next state after 2. Once, the program gets to state 62, or once IRD=3, the next state would be 29 in this case.