Ignacio Gonzalez

ieg356

Lab 4 Readme

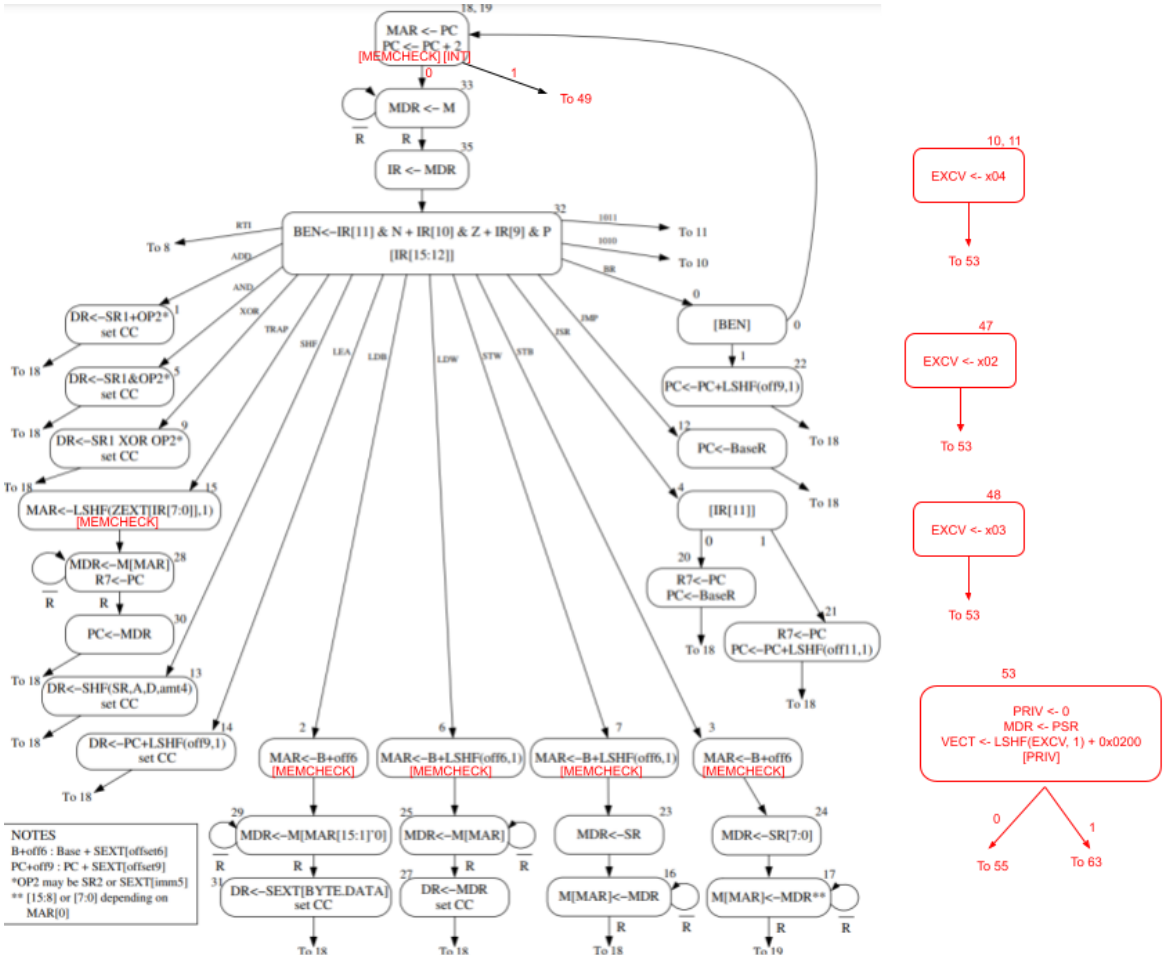
State Diagram

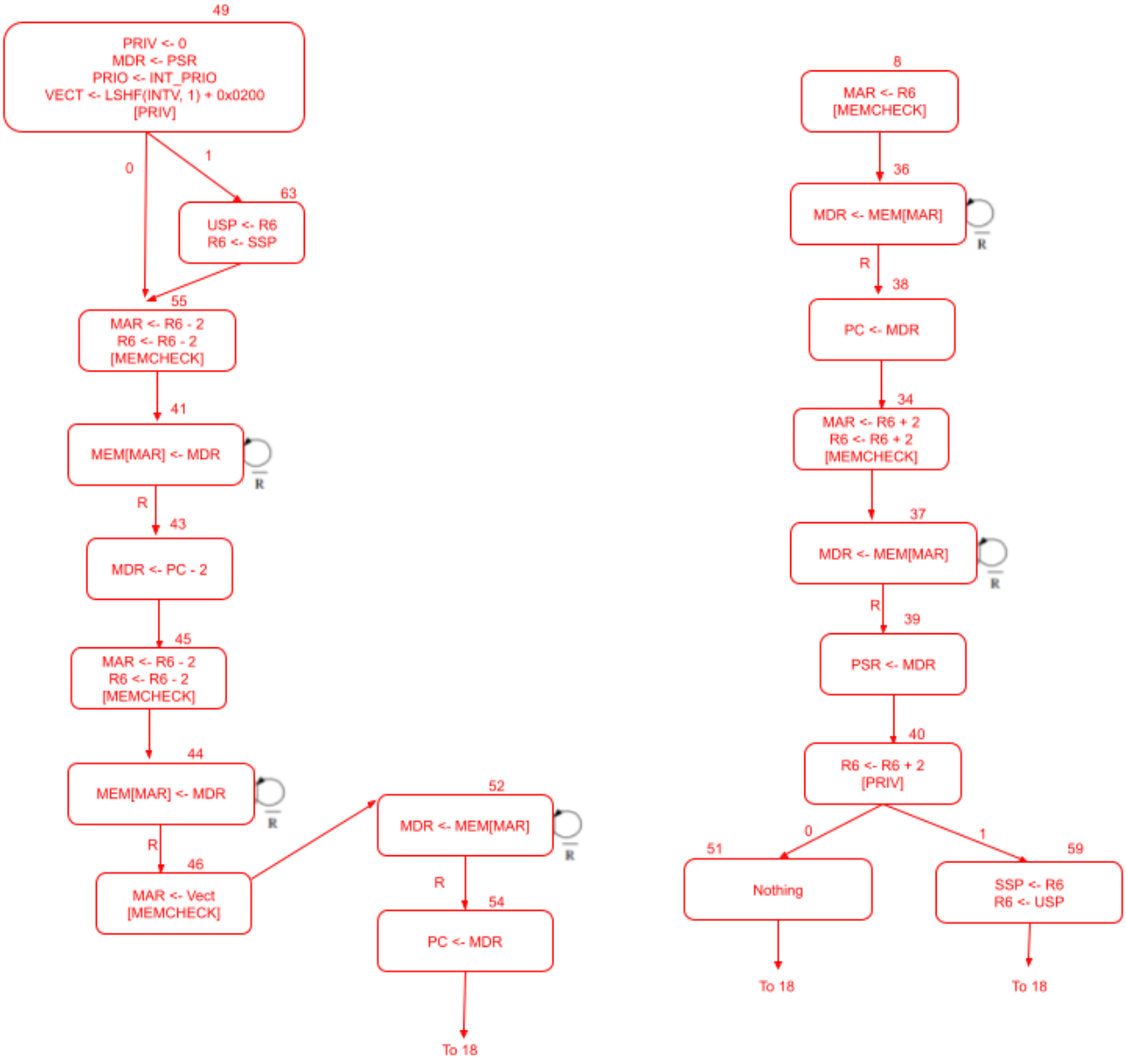
The [MEMCHECK] assertion is a unique control signal for the microsequencer that will move the processor into the exception handling process upon detection of any memory access exceptions. (See the microsequencer)

States 47 and 48 are for handling the memory exceptions, 10 and 11 handle illegal opcodes. 53 handles all the exceptions as well before branching into the interrupt process.

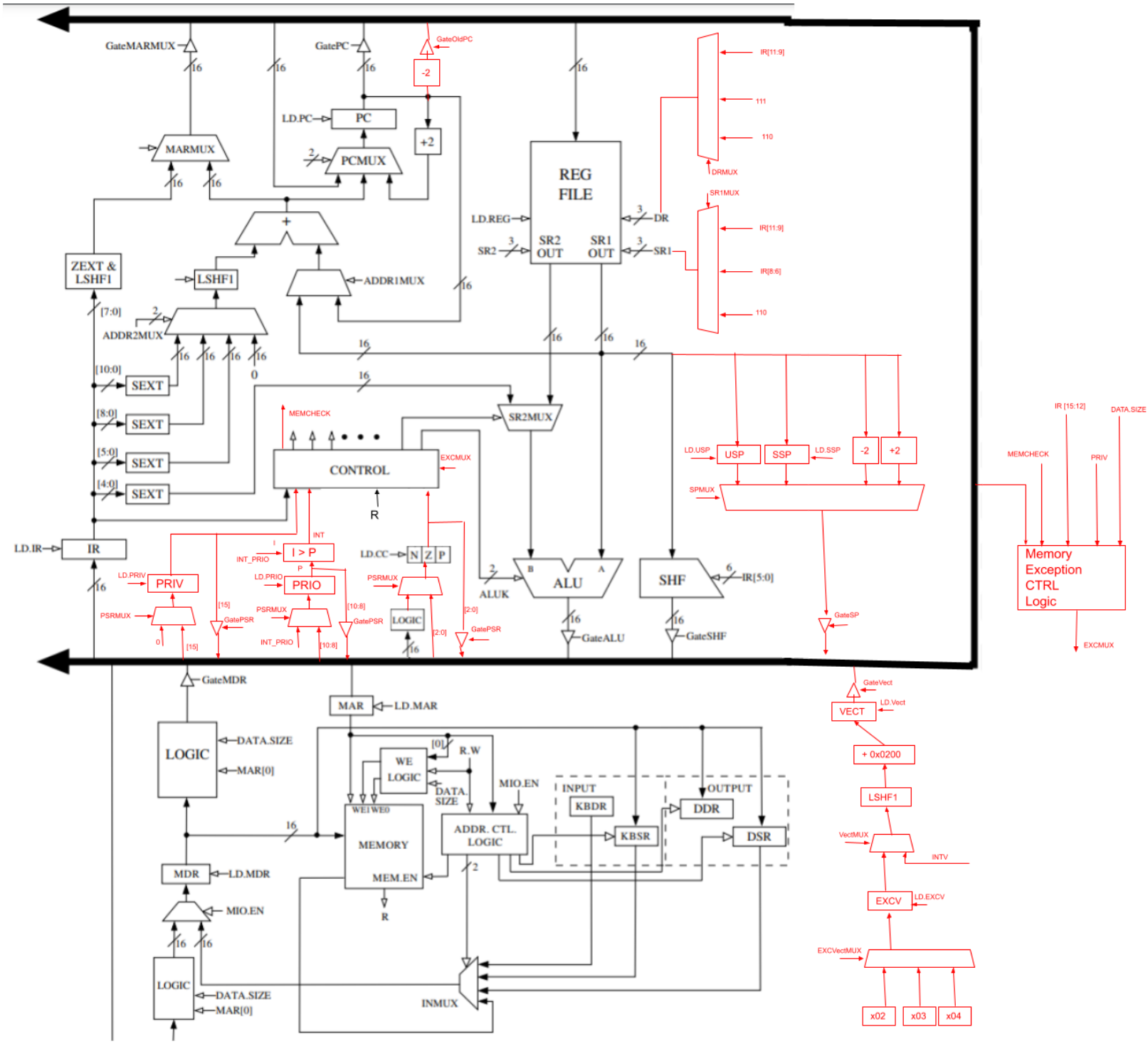
State 49 handles the interrupt process and state 55 is where the exceptions branch into the interrupt process.

State 8 handles RTI.




49
PRIV <- 0
MDR <- PSR
PRIO <- INT_PRIO
VECT <- LSHF(INTV, 1) + 0x0200
[PRIV]
0
1
63
USP <- R6
R6 <- SSP
55
MAR <- R6 - 2
R6 <- R6 - 2
[MEMCHECK]
41
MEM[MAR] <- MDR
R
R
43
MDR <- PC - 2
45
MAR <- R6 - 2
R6 <- R6 - 2
[MEMCHECK]
44
MEM[MAR] <- MDR
R
R
46
MAR <- Vect
[MEMCHECK]
52
MDR <- MEM[MAR]
R
R
54
PC <- MDR
To 18
8
MAR <- R6
[MEMCHECK]
36
MDR <- MEM[MAR]
R
R
38
PC <- MDR
34
MAR <- R6 + 2
R6 <- R6 + 2
[MEMCHECK]
37
MDR <- MEM[MAR]
R
R
39
PSR <- MDR
40
R6 <- R6 + 2
[PRIV]
0
1
51
Nothing
To 18
59
SSP <- R6
R6 <- USP
To 18

Datapath



- The PRIV reg is for PSR[15].
- PRIO reg is for PSR[10:8]

- INT_EN holds the interrupt enable bit
- USP holds the user stack pointer when leaving a user process and entering system process
- SSP holds the system stack pointer when leaving a system process and entering user process
- The Memory Exception Control Logic takes the MAR off the bus (when MAR is being loaded). The MEMCHECK control signal tells the logic block to check for any memory exceptions; when MEMCHECK is not asserted, it will output 0 to continue with the next state normally. This logic checks for a user privilege exception using the MAR, PRIV, and IR[15:12]. The opcode is used to determine if the current execution is of a trap code, in which case the user can access the trap vector table in system memory. It also checks for unaligned memory access using DATA.SIZE and MAR.
- Vect reg stores the vector that an interrupt or exception will jump to
- EXCV reg stores the current exception vector

Control Signals

- LD.PRIV - loads bit 15 from BUS into PSR
- LD.PRIO - loads bits [10:8] from BUS into PSR
- PSRMUX - controls whether to load psr with specific values or to load stored PSR from the BUS
- GatePSR - gates PSR onto BUS
- LD.SSP - loads R6 into SSP register
- LD.USP - loads R6 into USP register
- SPMUX - selects whether to output R6 + 2, R6 - 2, SSP, or USP
- GateSP - gates output from SPMUX onto BUS
- GateOldPC - gates PC - 2 onto BUS
- LD.Vect - loads shifted intv/excv + 0x0200 into reg
- GateVect - loads vect reg onto BUS
- VectMUX - selects whether to load intv or excv

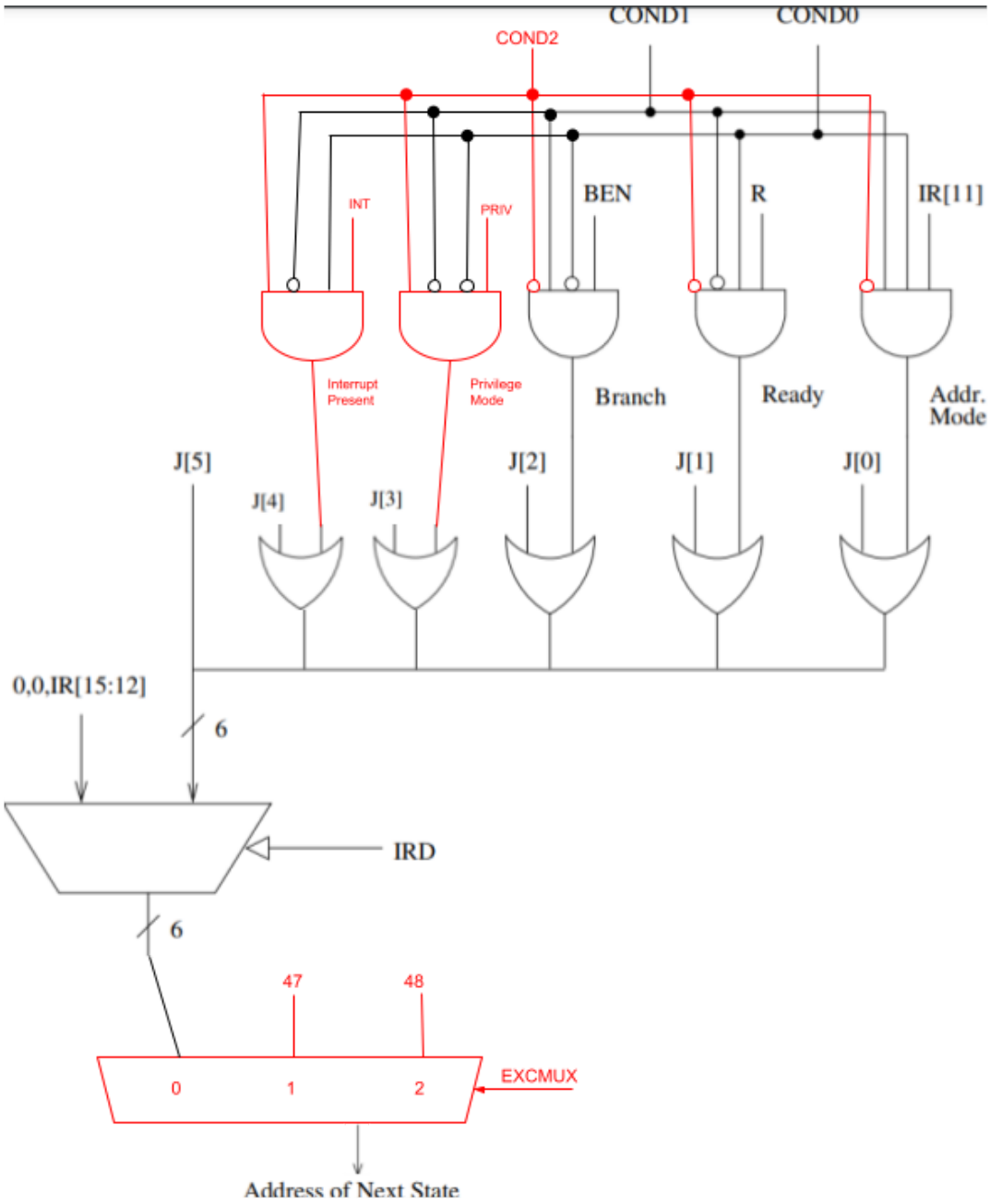
- LD.EXCV - loads the exception vect into the excv reg
- EXCVectMux - selects what the exception vect is
- LD.INT_EN - loads the interrupt enable bit
- INT_ENMUX - selects what the new interrupt enable bit is

MicroSequencer

The interrupt present mode is checked at the beginning of state 18,19 to determine if there is a new interrupt that needs to be serviced. It will jump to state 49 to begin interrupt handling.

Privilege Mode is checked in the exception/interrupt sequences and RTI to update the R6 stack pointer to and from the USP/SSP.

EXCMUX is the control signal output from the Memory Exception Control Logic. When EXCMUX is 0, the normal microsequencer process is followed. When EXCMUX is 1, the microsequencer goes to state 47 to handle the memory access protection exception. When EXCMUX is 2, the microsequencer goes to state 48 to handle the unaligned memory access exception.


COND1
COND0
COND2
INT
PRIV
BEN
R
IR[11]
Interrupt Present
Privilege Mode
Branch
Ready
Addr. Mode
J[5]
J[4]
J[3]
J[2]
J[1]
J[0]
0,0,IR[15:12]
6
IRD
6
47
48
0
1
2
EXCMUX
Address of Next State