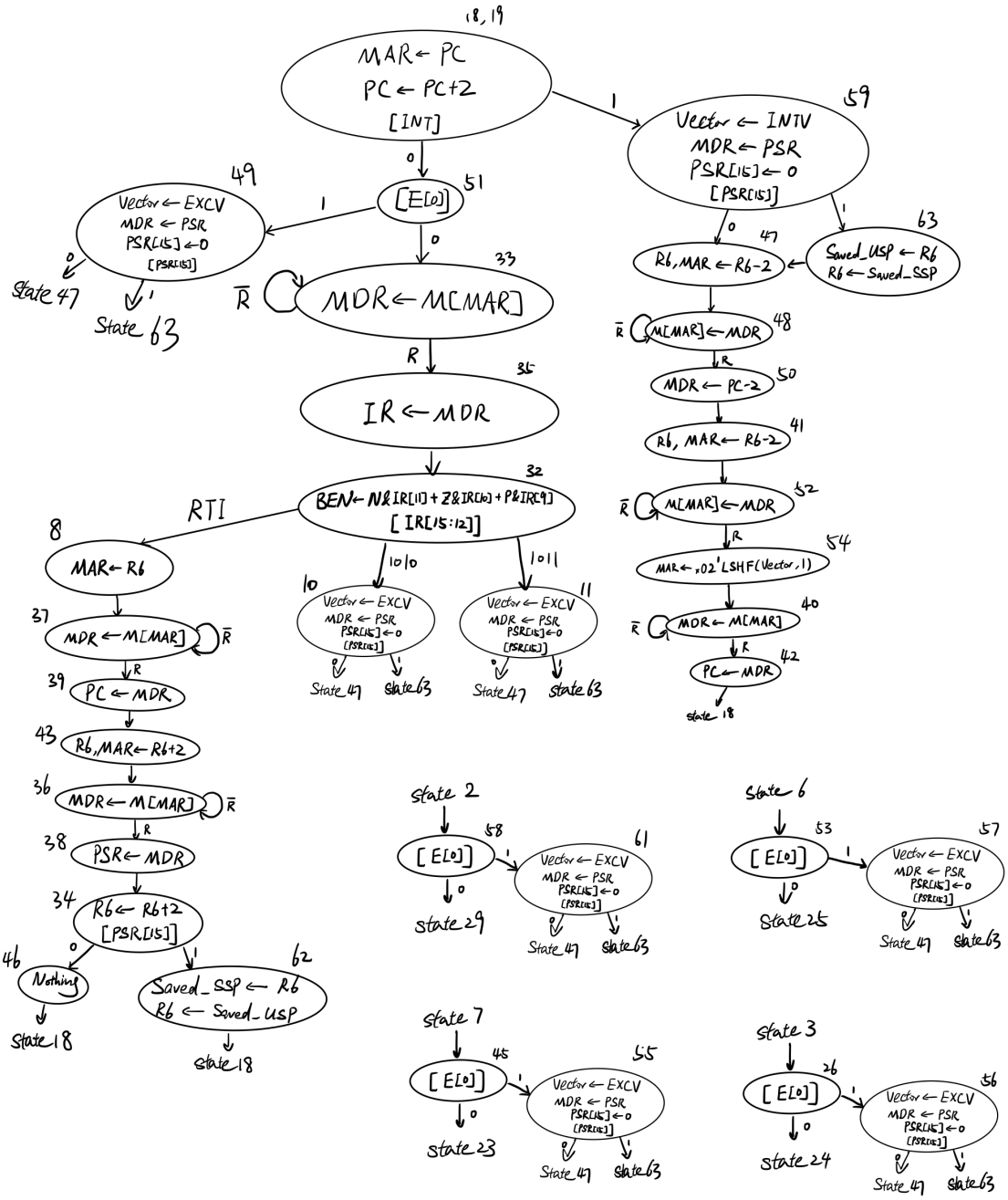
Xinyuan (Allen) Pan
Lab4 Documentation

**State Machine Modifications**




State 18 now checks the INT bit and determine if there is an interrupt request. INT is set by the interrupt requesting device. On the right side from state 59 to 42 are the interrupt/exception set up process. What they do is basically storing old PSR and PC to the supervisor stack and change PC to the starting address of the handler determined by the vector value. Also at state 59 the privilege bit is checked to determine whether the machine is currently in supervisor or user mode. If in user mode, we need to save R6 in Saved_USP and set it to SSP.
State 51 checks whether or not there is a protection exception or an unaligned access exception. I added this state here because JMP or JSRR instructions may change PC to an odd address or a supervisor-privileged address. Also, some subpar assemblers may allow putting the

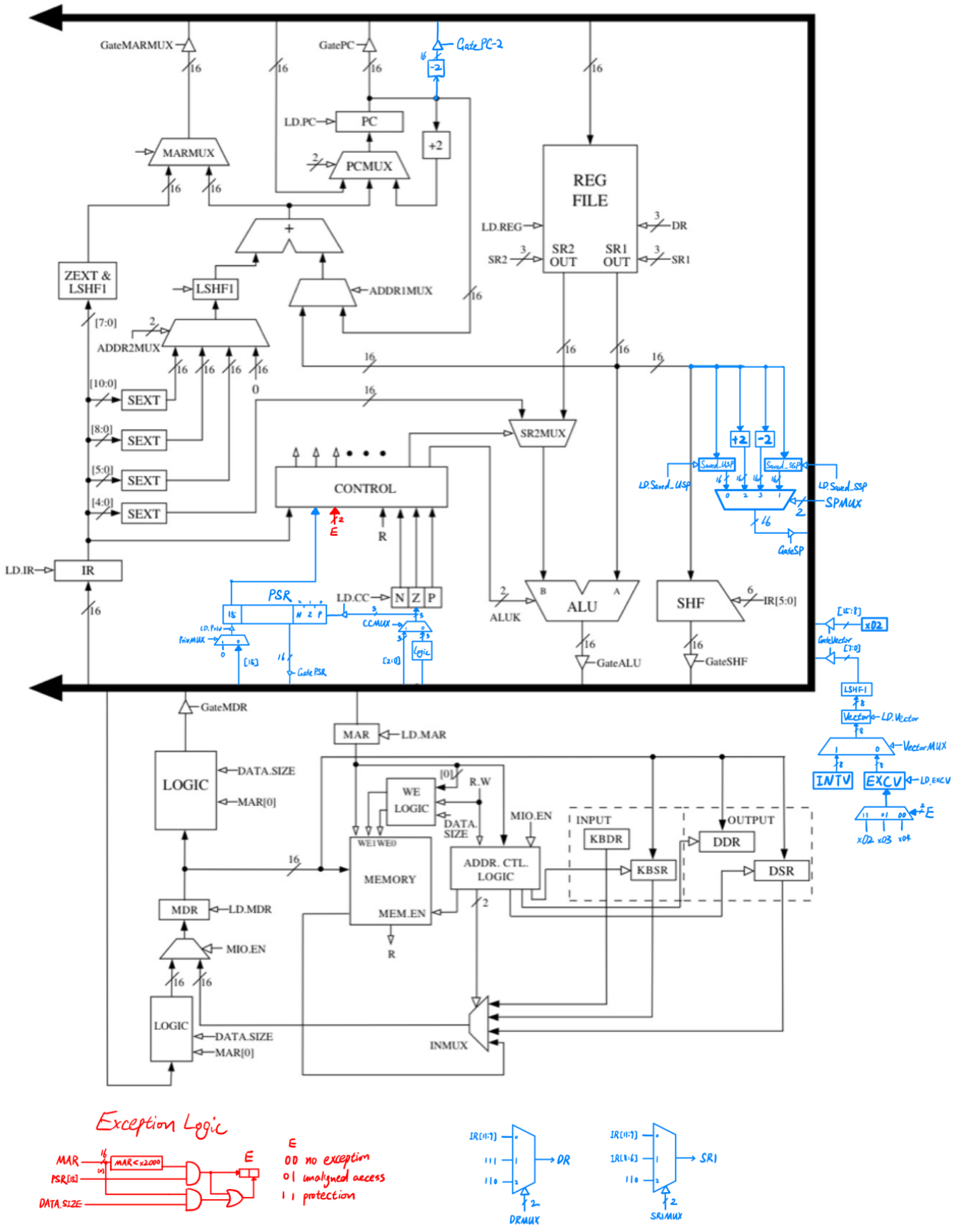
user program at the address below x2000. Thus I think state 51 is necessary. Details about the E register will be explained in datapath section. If there is an exception, then state 49 does the same thing as state 59, except that vector is loaded with the EXCV. And for the same reason, each store and load instruction needs to be checked, shown at the bottom.
States 10 and 11 are the two unknown opcodes, so they generate illegal opcode exception, doing the same thing as state 49.
The left side is the states for RTI. The first two values on the supervisor stack, the PC and PSR, are restored. This directs the program back to where we left for the interrupt/exception handling, and restores the architectural states. R6 is changed back to USP if the privilege bit shows it's in user mode.
The encoding of states are explained in the microsequencer section.

Since some of the old states now have new next states, the J and COND signals for them will be different.

**Data Path & Control Signals**
*Control Signals* (starting from 0):
SR1MUX: IR[11:9], IR[8:6], R6
DRMUX: IR[11:9], R7, R6
VectorMux: EXCV, INTV
SPMUX: Saved_USP, Saved_SSP, R6+2, R6-2
PrivMUX: BUS[15], 0
CCMUX: logic, BUS[0:2]
LD.EXCV: No, Yes
LD.Vector: No, Yes
LD.Saved_SSP: No, Yes
LD.Saved_USP: No, Yes
LD.Priv: No, Yes
GateVector: No, Yes
GateSP: No, Yes
GatePC-2: No, Yes
GatePSR: No, Yes

*Exception Logic*:
A 2-bit register E is determined by the logic shown in the datapath. It checks for protection and unaligned access exception.
00: no exception
01: unaligned access
11: protection
This E value is used as the select to the EXCV mux. 01 for unaligned access (x03), 11 for protection (x02), and 00 for illegal opcode(x04). 10 will never occur.
It's not as obvious that 00 is for illegal opcode. When the program is in state 10 or 11, it means there are no protection or unaligned access exceptions (would have been handled in state 51).

Xinyuan (Allen) Pan
Lab4 Documentation



Therefore, if the program reaches state 10 or 11, E must be 00 and Exception vector is x04. In other states, since LD.Vector is not asserted, it wouldn't matter which value EXCV takes.
And this implementation ensures that protection has higher priority than unaligned access, since if protection exception occurs, then no matter whether unaligned access occurs or not, E is 11.
E is sent to the control unit, and will be used in the microsequencer. E[0] is the condition used in state 51, 55, 56, 57, 61, since E[0]=0 means there is no exception.
Since E depends on MAR, PSR[15] and DATA.SIZE, the "[E[0]]" states always come after MAR is loaded with the appropriate value and DATA.SIZE signal must be set to the correct value .

VectorMux selects between interrupt vector and exception vector, and Ld.Vector loads it. The left shifted Vector is concatenated with x02 to form the address in vector table, and gated to the bus.
Also, LD.EXCV is asserted in every state that tests [E[0]], and the following state loads Vector, so every time we will get the correct exception vector. This is also one of the motivations behind the "[E[0]]" states in the state machine. This saves the number of states by combining protection and unaligned access exceptions together (without using a control signal as select).

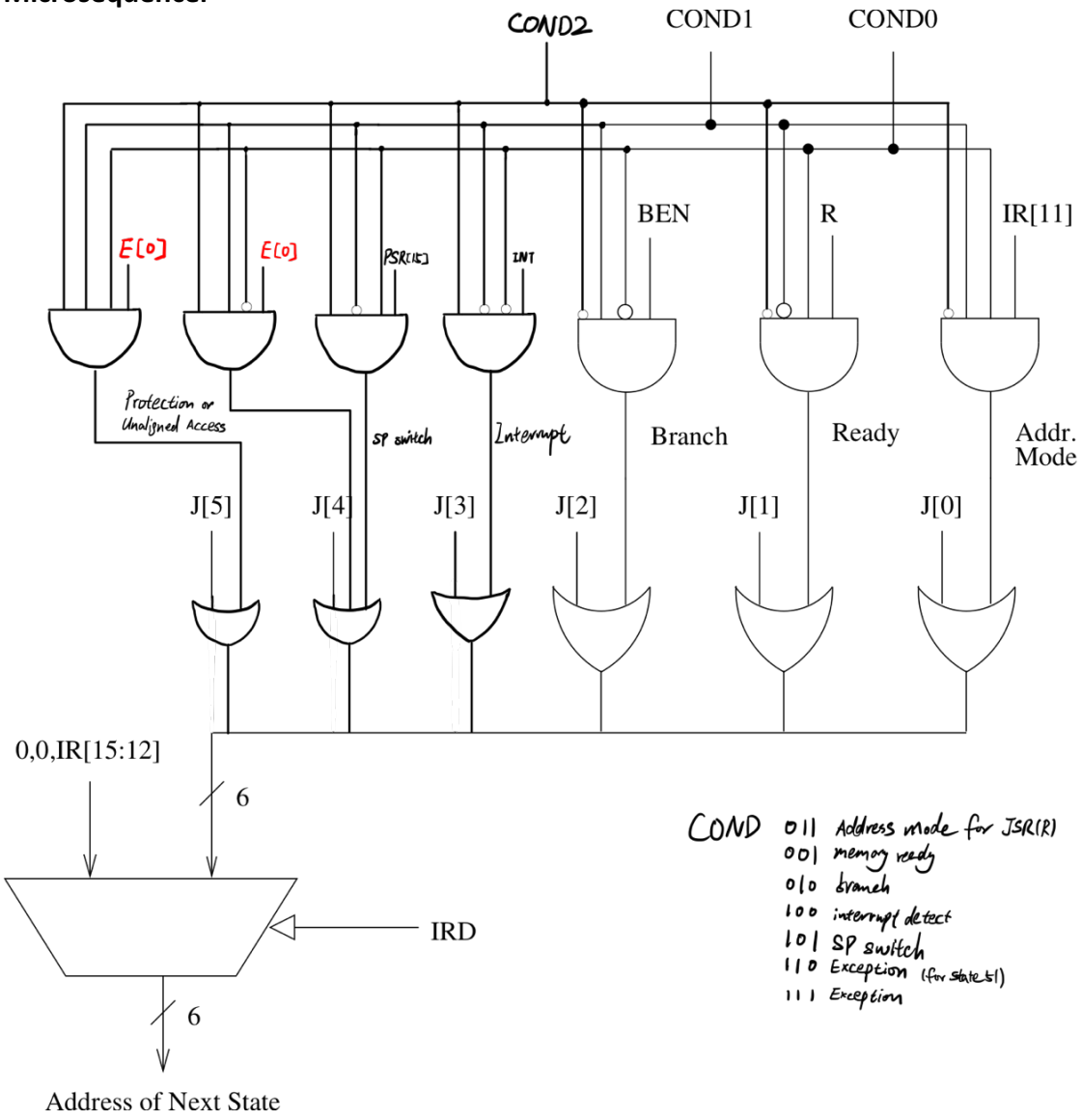
PSR has two parts that are in use. PSR[15] is the privilege mode bit, controlled by LD.Priv, and PSR[2:0] is basically a copy of the conditional codes. In state 38 we pop the PSR from the stack and restore it, so we need to load the privilege mode bit and the conditional codes. CCMUX and PrivMUX will choose the value from the bus. LD.CC also controls the bit 2:0 of PSR – every time conditional codes are set, the same values are copied to PSR[2:0] (CCMUX would choose the logic). GatePSR will pass the entire PSR to the Bus. And in states 10, 11, 49, 59, 55, 56, 57, 61, PrivMUX selects 0.

GatePC-2 simply passes the old PC to the bus, which is then pushed to the stack.

DRMUX and SR1MUX are expanded to support R6. When we deal with the stack pointer, we need a way to access R6 quickly. Two register, Saved_SSP and Saved_USP are used to store the respective stack pointers, controlled by LD.Saved_SSP and LD.Save_USP. Since R6 is used for both, we need a way to swap R6 between them. And the +2 and -2 are fast ways to decrement and increment stack pointer. SPMUX selects the desired stack pointer value, and GateSP pass it to the bus.

Xinyuan (Allen) Pan
Lab4 Documentation

**Microsequencer**




COND0: unconditional
COND1: memory ready
COND2: branch
COND3: address mode for JSR(R)
COND4: interrupt detect
COND5: Stack pointer switch
COND6: Exception (for state 51)
COND7: Exception (for other states)

COND is expanded to 3 bits since there are more conditions in state transitions.

Xinyuan (Allen) Pan
Lab4 Documentation

As before, J holds the value for the next state for unconditional states, holds its own state number for memory access states, holds the lower next state number for other states.

COND4 tests for interrupt signal. At state 18, if the INT signal is asserted, then the next state would be 59 instead of 51 (apart by 8).

COND5 tests for the privilege mode. This is mainly for the swap of R6 between SSP and USP. Two states followed are apart by 16.

COND6 and COND7 both use E[0] to determine next state, which indicates whether or not there is an exception. Thus, the next states of the "[E[0]]" states need to have state numbers apart by 16 or 32.

Most of the purposes are straightforward, except for COND6 and COND7. I had to use both COND6 and COND7 for Exception because state 33 (100001) cannot add 32. In my implementation, states that read "[E[0]]" have to be followed by two states apart by 32 (COND7). Thus I have to use COND6 for state 51, which has offset 16 (33 to 49).