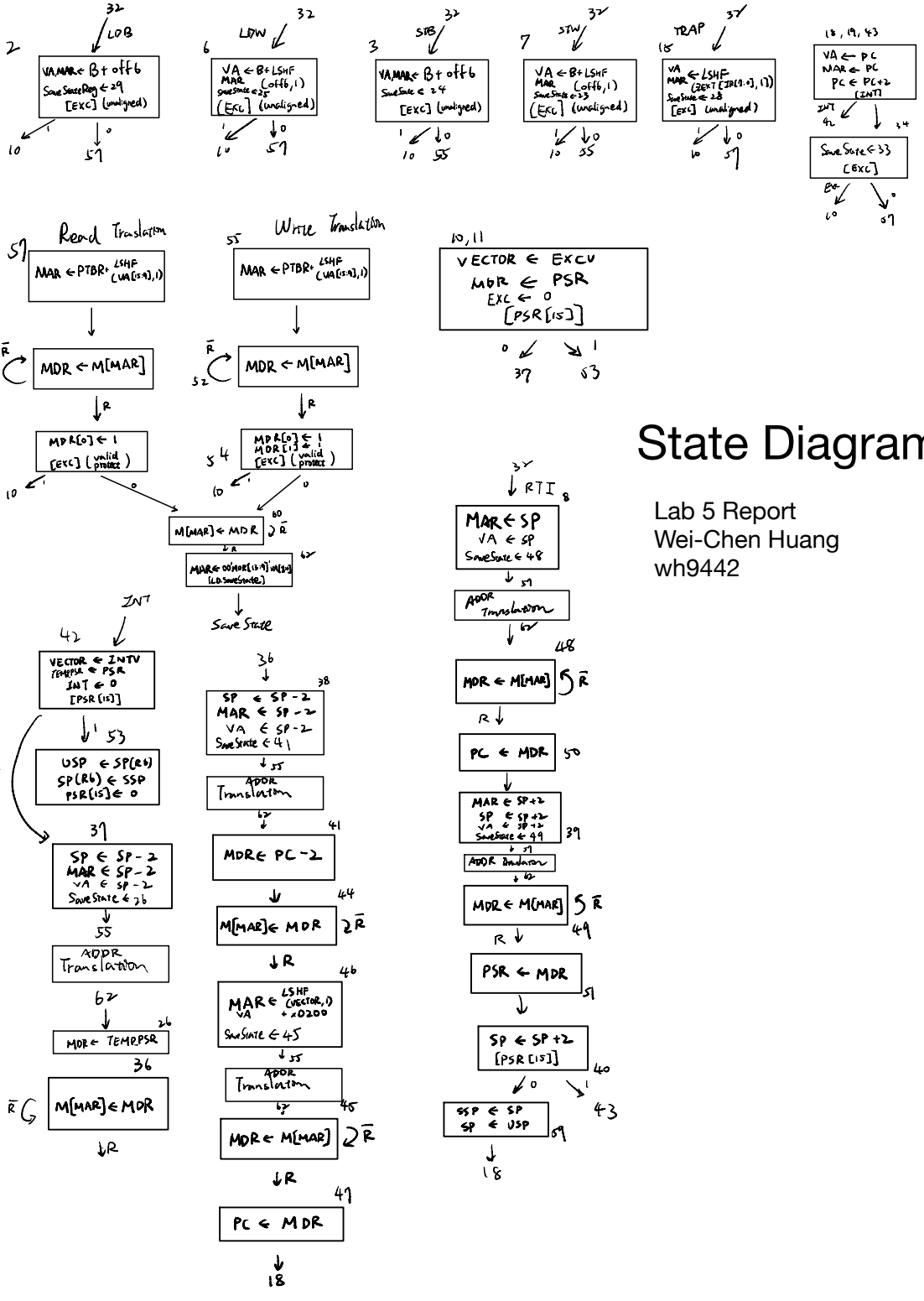


State Diagram

Lab 5 Report
Wei-Chen Huang
wh9442

Data Path

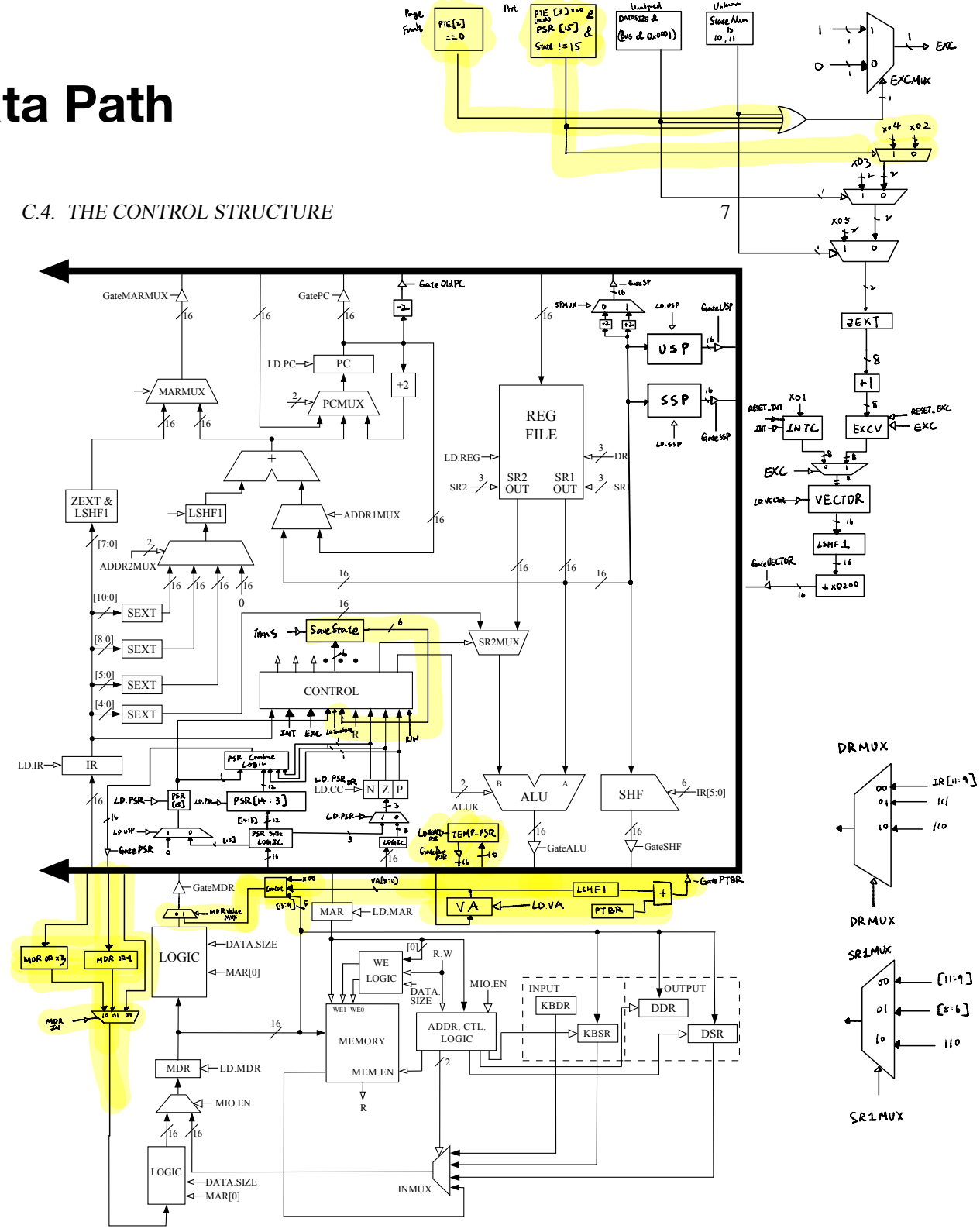
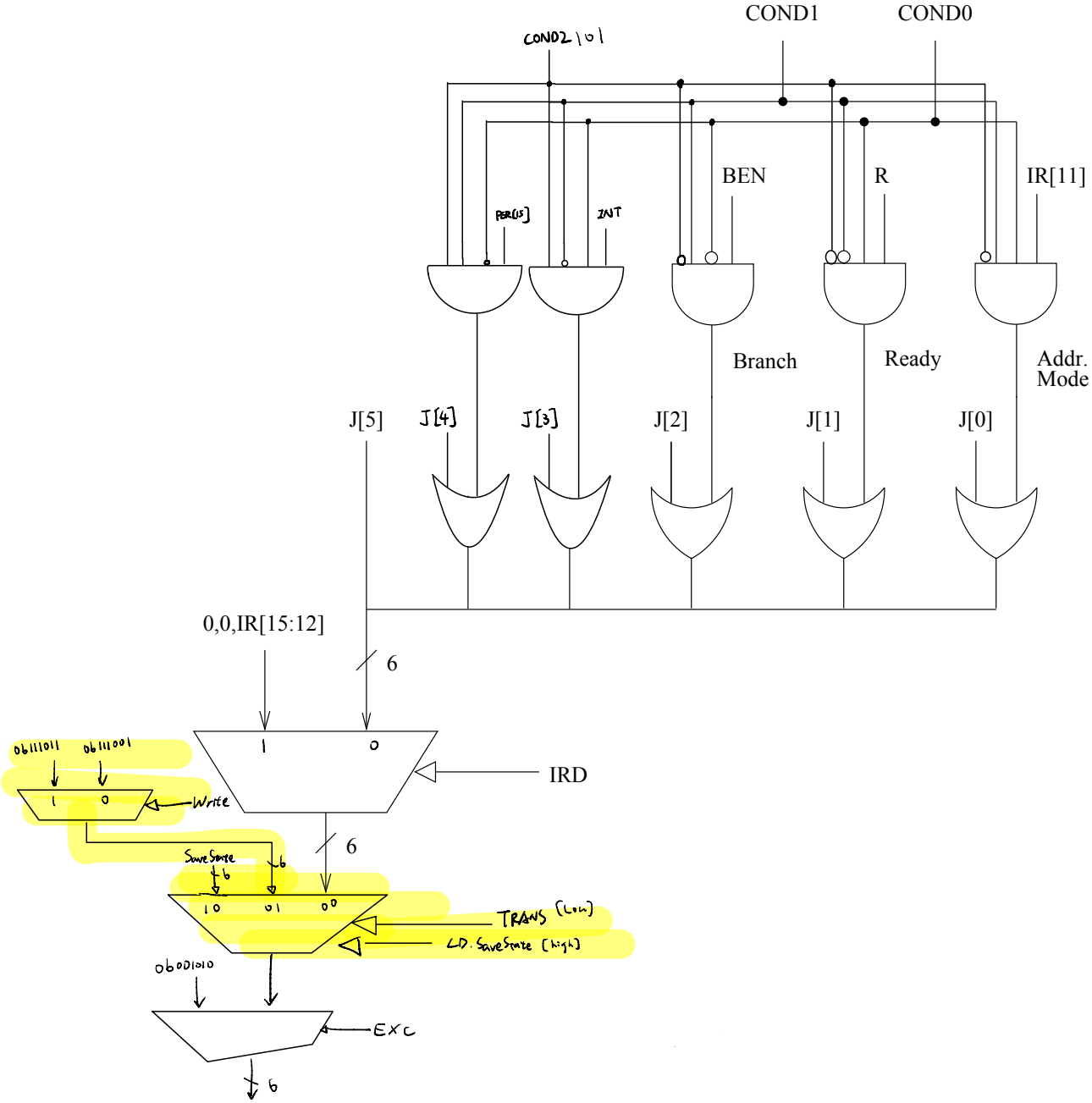


Figure C.3: The LC-3b data path

provide you with the additional flexibility of more states, so we have selected a control store consisting of 2^6 locations.

Microsequencer



State Machine Change

I add a **translation logic** to the state machine. All the states that use VA will use this translation logic to get PA and stored back to MAR. Then, we could access the value in the memory.

The translation starts from state 57 (Read) /55 (Write). We stored the PTBR + LSHF(Virtual Page Number, 1) back to MAR. In State 56/52, we get the PTE of VA from memory. In state 58 (read operation), we need to check exceptions for protection exceptions and page fault here. If an exception happens, the next state would be state 10 and EXC would be assigned a designated value according to the type of exception. On the other hand, if there is no exception, we will modify the PTE's reference bit to 1 and go to state 60. For write operation, we will go to state 54 instead of state 58. In addition to the steps in state 58, we need to modify the dirty bit in the PTE. In state 60, we store the modified PTE back to memory. In state 62, we send the PA by concatenating zeros, page frame number and offset from VA back to MAR for later usage. The last step,we need to assert the LD.SaveState signal to let the microsequencer load the next state from the SaveState Register to return to the original process before translation.

For all the blocks that access memory using the VA, we need to store the next state in the original state diagram to SaveStateReg by the signal of TRANS, and still retain the exception check of unaligned exceptions at the original state blocks. If exceptions happen, we go to state 10; if not, we go to state 57 to get the translated address. The translation blocks are also inserted into the process of exception handling.

Datapath Change

I added several changes highlighted on the graph based on the lab 4 version.

TRANS, SaveStateReg
Add SaveState register to save the next state to execute after translation, and we use the TRANS signal to control load or not. Also, SaveState will go into the microsequencer.

TEMP_PSR, LD.TEMPPSR, GateTEMPPSR
Add the register and two signals to control the storing of PSR. Since we modify the state diagram and MDR will be overridden. We cannot put PSR to store into MDR before translation. Instead, we use TEMP_PSR to store it. After translation, we get value from the register and put into MDR to continue the exception handling.

MDRINMUX
In the MDRINMUX, we choose the value for the MDR. In state 58/54, the next MDR is equal to MDR OR 3 or MDR OR 1. So we use the MDRINMUX to control the value loaded into MDR by 2 bits.

VA, LD.VA
We store VA from BUS for later translation.

PTBR, GatePTBR
In this datapath, we will get VA value and left shift 1 and ADD with the value in PTBR. The final value is the address of PTE and we use GatePTBR to control it.

Concat LOGIC, MDRValueMUX
Once we get the PTE and stored into MDR, in the next state, we will evaluate the PA of the VA. We use concat logic to get the physical address. We use MDRValueMux to select the output. If MDRValueMux is high, we will send the PA to the bus.

Page Fault, Protection Exception Detector
We add and modify two blocks to detect exceptions and generate corresponsive signals for EXC to load.

Microsequencer Change

Based on lab 4, TRANS, and Read/Write signal is added to the microsequencer. If TRANs is asserted, the next state will go to 57 (read)/ 55 (write). If LD.SaveStae is asserted, we will load the next state from the SaveState Register. The rest of them are unchanged.

Modified Control Signal

LD.VA	NO, LOAD
LD.TEMPPSR	NO, LOAD
LD.SaveState	NO, LOAD
GateTEMPPSR	NO, YES
GatePTBR	NO, YES

MDRINMUX

[00] - BUS value
[01] - MDR OR x1
[10] - MDR OR x3

MDRVALUEMUX

[0] - LOGIC of MDR
[1] - Physical Memory computed value

TRANS	NO, YES