**Datapath and control signals**

**Labeled all the structures on the datapath**

My first structure from my datapath that I added is the structure that is responsible for creating the address based on an offset and a base for both interrupts and exceptions. One state that uses this structure is state 42 where MAR is populated with the adder from this structure which adds the offset based on what exception it is or interrupt plus the base value of 0x0200. The point of this structure is to yield the address of the memory location which contains the value of the starting address of the service routine that the program is trying to go to based on the exception or interrupt.

The control signals added to the first structure are VectorMux, LD. Vector and gateTable. VectorMux is used to figure out what the vector is based on if there is an interrupt or exception. There are three exceptions and one interrupt thus 4 inputs and 2 bits to choose the value of the vector. Before the values output out of the mux the value is left shifted by one. Based on the value of LD. Vector, the output from the VectorMux either gets stored in the Vector register or not. Lastly the gateTable outputs the value to the bus from the adder output of 0x0200 and the Vector register value.

My second structure from the datapath is a structure I added so that exceptions and interrupts can change the state if needed. For instance, if the Exception bit was a one then the program goes to state 54, while if the interrupt bit was a one then the program goes to state 38. The other functionality of the second structure is to add the PSR register and create a way for the PSR to both output to the PSR and change its value.

The control signals from the second structure are [E], [I], PSRMux, LD.PSR, and gatePSR.

The exception and interrupt control signal are set based on the logic structure as seen in the datapath- this control signal is responsible for identifying specific exceptions and interrupt based on “logic” as seen in the datapath. The PSRMux is used to select whether you want the output of the PSR to be changing bit 15 of the PSR to 0 or to get the value from the bus. This value from the mux is loaded into PSR register based on the value of LD.PSR (if one then the value gets loaded into the register and vice versa). Lastly, gatePSR is used for outputting the PSR register value to the bus.

My third structure is not a new structure from the datapath but rather a modification of the DRMux. DRMux is used for selecting what register you want the destination be. I added Register six (R6) as a destination since R6 acts as a stack pointer and thus you need a way to store a value into R6. This modification helps the fifth structure from the datapath

The control signal DRMux changed from a one-bit selector to a two-bit selector since the number of inputs went from two to three. Thus, you need another bit selector for choosing between IR [11:9], R7, and R6.

My fourth structure is not a new structure from the datapath but rather a modification of the SRMux. SRMux is used for selecting what you want the source register to be (designated as SR1 out on the datapath). I added Register six (R6) as a source register since R6 acts as a stack pointer thus you need a way to get the value from R6 at all times. This modification helps the fifth structure from the datapath.

The control signals SRMux changed from a one-bit selector to a two-bit selector since the number of inputs went from two to three. Thus, you need another bit selector for choosing between IR [11:9], IR [8:6], and now R6.

My fifth structure is not a new structure from the datapath but rather a modification of the PCMUX. PCMUX is used for selecting what value you want PC to be loaded with or you want on the bus. In this instant, I modified the PCMUX so that the value of PC-2 could be outputted to the bus. Since PCMUX already has a two-bit selector, I made PC-2 as the fourth input for the four to one mux. This modification is needed for a state such as 39.

My sixth structure from the datapath is a structure that moves the value of SR1out to USP and moves the value of SSP to SR1out or moves the value of SR1out to SSP and moves the value of USP to SR1out depending on the state. The structure is also responsible for either incrementing or decrementing the value from SR1out by 2. In the states that I added, the value of SR1out is usually R6 as that acts as the stack pointer throughout the program. This structure is made possible by the modifications made in 3 and 4 as you need the value from R6 and you need to output to R6 as well whenever this structure is in use.

The control signals used in the sixth structure is LD.SSP, LD. USP, SPMUX, and GateSP. The control signal LD.SSP and LD.USP are used to load the value from SR1Out (R6 in all my states). This can be seen in state 44 and 57. The control signal SPMUX is used to select what value you want outputted out of the mux. In this case it could either be USP register, SR1out+2, Sr1out-2, or SSP register. Lastly the value from the mux is only outputted if the value of GateSP is one.