Computer Science 605.611

Problem Set 5

1. Suppose that our MIPS core instruction subset (add, sub, slt, and, or, lw, sw, beq & j) is expanded to include another instruction, the or immediate (ori) instruction. Answer each of the following questions about possible modifications to the multi-cycle datapath and FSM that may be required to support this additional instruction.

a) (3) What is the minimum number of new states required for the FSM (finite state machine) description of the ori instruction execution?

I would add 2 new states to the current 10 state FSM. The ori instruction is an I-type instruction, for which there is currently no path in the 10 state FSM from module 5. I would add an Op = I-type execution and an I-type completion state. To execute an I-type instruction, then, we would have to pass through 4 states, first the instruction fetch state, then the instruction decode/register fetch state, then the I-type execution, and finally the I-type completion before returning back to state 0.

I would add 1 new state to the current 10 state FSM. The ori instruction is an I-type instruction, for which there is currently no path in the 10 state FSM from module 5. I would add an I-type execution state below what is currently state 1. To execute an I-type instruction, then, we would have to pass through 3 states, first the instruction fetch state, then the instruction decode/register fetch state, then the I-type execution, before returning back to state 0.

b) (3) Identify any new control bits that the control unit has to output for the ori instruction.

To handle the instruction decode / register fetch logic, a new bit would have to be added to signify what is now a 5-way choice from state 1.

c) (3) For each new state required for ori, list all control bit settings used in the new state and indicate what determines when to transition into the new state.

For the I-type execution state:
ALUSrcA = 1
ALUScrB = 100 (ALUSrcB is now 3 bits long since there is a 5-way choice from state 1)
ALUOp = 11

d) (3) Identify any new hardware devices that are required in the datapath to support the execution of the ori instruction.

Since the ALUScrB is now 3 bits instead of 2, we will need to change the 4 bit multiplexer for ALUSrcB to a 5 bit multiplexer. We will also have to add microcode to the ALU to run the ori instruction.

2. The micro-programmed implementation of our MIPS multi-cycle datapath employs a micro-PC.
a) (3) How many bits does the micro-PC contain ?

Our micro-PC contains 4 bits, representing

b) (3) What do the bits within the micro-PC indicate?

The bits within the micro-PC indicate the next state in the finite state machine. The states are numbered in binary (0 -> 0000, 9 -> 1001), thus the micro-PC can control the flow through the various states in the FSM.

c) (3) When the micro-PC is incremented, by how much is it incremented?

For states 0, 2, 3, and 6, the state is incremented by 1, to states 1, 3, 4, and 7 respectively. For states 4, 5, 7, 8, and 9, the state is incremented back to the 0 state. The other states, state 1 and 2, make a decision based on the opcode bits, which control whether state 1 goes to state 2, 6, 8 or 9, and whether state 2 goes to state 3 or 5.

d) (3) When the micro-PC is updated, what bits within the micro-instruction indicate how to update the micro-PC?

The state and output of the micro-PC are determined by the opcode and the previous state. Since the current state of the micro-PC is not indicated in the micro-instruction, the bits within the micro-instruction which indicate how to update the micro-PC are the opcode bits.

3. Both of our MIPS single-cycle and multi-cycle datapaths use control bits to indicate the operations and direct the actions that take place when executing machine instructions.
a) (3) How many control bits are required for our MIPS single-cycle datapath?

The opcode for MIPS single-cycle datapath is 6 bits long.

b) (3) How many control bits are required for our MIPS multi-cycle datapath?

The AddrCtrl used for multi-cycle datapaths is 2 bits long.

4. If the control unit for the single-cycle datapath were implemented using a PLA (programmable logic array), what is the minimum number of inputs and the minimum number of outputs required for the PLA? Name each of the inputs and the outputs.

a) (5) minimum inputs are: The opcode bits: Op0, Op1, Op2, Op3, Op4, Op5 (6 total inputs)

b) (5) minimum outputs are: RegDst, ALUrc, MemtoReg, RegWrite, MemRead, MemWrite, Branch, ALUOp1, ALUOp2 (9 total outputs)

5. a) (3) Why do CISC processors tend to use micro-programming?

Micro-programming has greater flexibility, allowing us to simply change the content of the control ROM to alter the behavior of the machine instruction. Using micro-programming is often the only viable approach to implementing complex instructions.

b) (3) Why do RISC processors not use micro-programming?

RISC instructions are typically so simple, that it is easy to hard-wire the instructions, which is significantly faster than retrieving and carrying out micro-instructions.

c) (3) What is the main reason that ROM rather than RAM is used to store micro-code (i.e., micro-programs)?

ROM can hold data without power, while RAM cannot. Thus, micro-code which is written in to ROM can be retrieved even after a computer is shut off or runs out of battery. This can be useful for writing firmware, and other applications.

6. Answer each of the following questions about the micro-programmed implementation of our MIPS multi-cycle datapath described in module 5:

a) (3) How many bits are contained within each machine instruction?

Just as with the single-cycle MIPS datapath, there are 32 bits contained in each machine instruction.

b) (3) How many bits are contained within each micro-instruction?

Each micro-instruction within our core MIPS instruction sheet and based on the implementation in module 5 has 18 bits.

c) (3) What is the total number of micro-instructions contained in the control memory for the micro-programmed system for our MIPS core instruction subset?

There are 10 micro-instructions contained in control memory for the micro-programmed system for the MIPS core instruction sheet, one instruction for each state in the FSM.

d) (5) Use hex to show each micro-instruction in the sequence of micro-instructions (i.e., the micro-program) needed for the add machine instruction.

add goes from state 0 -> 1 -> 6 -> 7
Micro-instruction state 0: 1001 0100 0000 1000
Micro-instruction state 1: 0000 0000 0001 1000
Micro-instruction state 6: 0000 0000 0100 0100
Micro-instruction state 7: 0000 0000 0000 0011
Converting to hex:
9408
0018
0044
0003

7. Answer the following questions about the finite state machine (FSM) model of our muti-cycle datapath control shown in module 5:

a) (3) How many separate states are there in the FSM?

10 states, numbered 0-9.

b) (3) How is the opcode within each machine instruction used by the FSM?

The opcode in each machine instruction is used by states 1 and 2 to determine what state to advance to next. State 1 can advance to state 2, 6, 8, and 9 based on whether the opcode is R-type, BEQ, J, or lw/sw. State 2 can advance to states 3 or 5 depending whether the opcode is low or sw.

c) (3) What is the maximum number of states used in the FSM for any of the instructions within our MIPS core instruction subset?

The maximum number of states used in the FSM is for the lw instruction, which requires 5 separate states, states 0, 1, 2, 3, and 4.

d) (3) Is the FSM an example of a Mealy machine or an example of a Moore machine (as defined in module 5 CU_implementation)?

Moore machines are characterized by having the output depend only on the current state. Mealy machines are characterized as allowing both the input and current state to determine the output. The 16 bit output produced by our FSM is determined solely by the current state, therefore the FSM in module 5 is a Moore machine.

8. The ALU shown in the multi-cycle datapath diagram in module 5 has two data input ports: the upper A input port and the lower B input port.

a) (5) Name each of the possible A inputs to the ALU?

The upper port of the ALU is a 2 bit multiplexer, which takes the program counter and the rs register as inputs. The multiplexer is controlled by the ALUScrA which decides whether the multiplexer passes the program counter or the rs register stored in latch A to the ALU.

b) (5) Name each of the possible B inputs to the ALU?

The lower port of the ALU is a 4 bit multiplexer, which takes the rt register, the constant 4 (if we want to increment the program counter), the sign extended displacement shifted left two bits (useful for computing branch target address), and the sign extended immediate (useful for computing memory addresses). The multiplexer is controlled by the 2-bit ALUScrB which determines which value to pass through.

9. (5) Our MIPS ALU outputs a single-bit flag called "zero" that is set to 1 to indicate when the ALU result is 0. Suppose that the ALU also output three other flags which, when set, indicate one of the following conditions:

- N to indicate a negative result
- V to indicate an arithmetic overflow
- C to indicate a carry

After the instruction sub $t0,$t1,$t2 subtracts $t2 from $t1 and places the result into $t0, some combination of the flags (Z, N, V, C) indicates that $t1 contained a value less than that in $t2? Write down the logic expression that shows the required combination of Z, N, V and C in simplest form.

$t2 < $t1 if:
C AND N OR
C AND N AND V
V and !N

Logic expression = (C * N) + (C * N * V) + (V * 'N)

10. You know that every instruction takes one clock cycle on the single-cycle datapath. How many clock cycles are required to fetch and execute each of the following instructions on our MIPS multi-cycle datapath?
a) (2) sw

sw uses states 0, 1, 2, and 5, therefore sw requires 4 clock cycles

b) (2) add

add is R-type
add uses states 0, 1, 6, and 7, therefore add requires 4 clock cycles

c) (2) slt

slt is R-type
slt uses states 0, 1, 6, and 7, therefore add requires 4 clock cycles

d) (2) lw

lw uses states 0, 1, 2, 3, and 4, therefore lw requires 5 clock cycles

e) (2) beq

beq uses states 0, 1, and 8, therefore beq requires 3 clock cycles