CSE30321 Computer Architecture I Fall 2009
Homework 02: Programmable Processors (100 Points)
Assigned: September 8 Due: September 15

---

1. **(40 Points)**

   Often, the best way to become familiar with a processor architecture is to modify or extend it in some way. This question will ask you to extend Vahid's six-instruction processor to add a new instruction by modifying the processor datapath and controller state diagram. This type of problem, which will recur throughout the homeworks, is also good practice for the final project.

   For this question, you need to modify the datapath of the 6-instruction processor (shown in Fig. 8.12, p. 436 in Vahid as well as in the lecture notes). The instruction should load a piece of data from memory into a register. However, unlike the previous load instruction, this time the address to load from *will be stored in a register*. The instruction mnemonic, encoding, and RTL are shown below:

   - MOV Rx, Ry
   - 1101 (opcode) – XXXX (Rx - destination) – YYYY (Ry - address source) – ZZZZ (unused/not needed)
   - Rx ← data memory(Ry)
   - Note: Just to be clear, the address sent to memory is the lowest 8 bits of the data that is in Ry.

   You'll need to describe the changes that you made to datapath (using both an illustration and a few short sentences). Also, show all necessary updates to the finite state machine. For your convenience (and ours!) a copy of the 6-instruction FSM is included in this assignment and can also be found on the course website.

   Finally, it is worth noting that this type of instruction has significant utility and is found in most modern microprocessors. For example, imagine that you wanted to stride through elements in memory – e.g. load data words d(0) through d(N) where N is a variable. Sending an address to memory that is in a register makes it very easy to update the address itself. If this instruction were not present, how would you handle a stride through memory? (It can be done with the 6 instructions initially associated with the 6-instruction processor.) Quantitatively, how many cycles would this new instruction save?

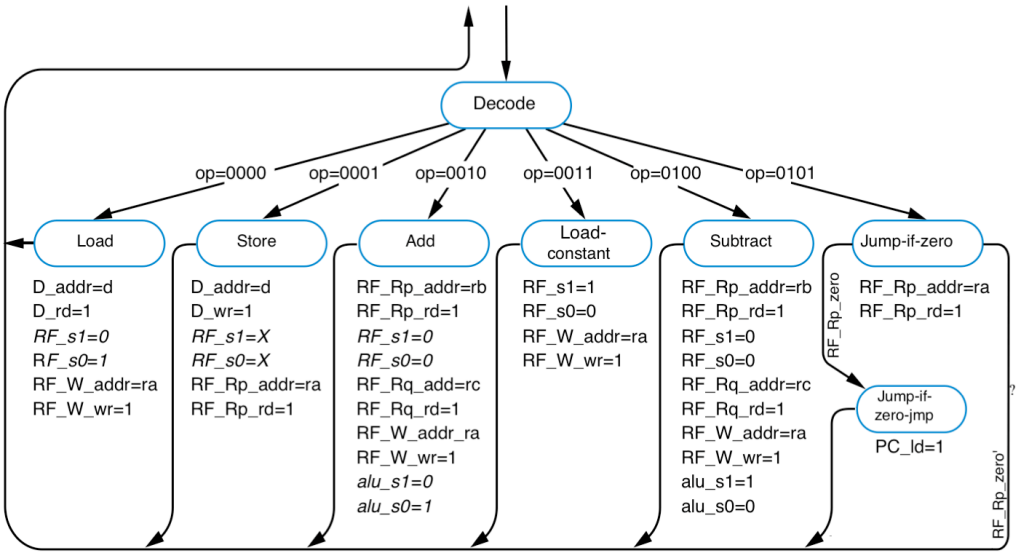

Figure 1: 6 Instruction Processor FSM (starting point for Question 1).

2. **(10 Points)**

   Add an unconditional `JMP` instruction to the six-instruction processor. The instruction should use opcode 0111 (binary), and the lower 8 bits of the instruction should be loaded directly into `PC`. That is, the jump address is absolute, not relative to the current `PC`. The highest order bits of `PC` can be set to zero.

   Again, you may obtain a copy of the datapath and control diagrams for the processor on the course website, or as Figures 8.12 and 8.13 in the textbook. You only need to show modifications to the diagrams. Be sure to clearly indicate the changes that you make, and briefly explain what you did and why you chose your particular implementation. Note that you should start with the original 6-instruction processor datapath – and not the datapath that you modified for Question 1.

3. **(30 Points)**

   This problem references the following code segment, written for the six-instruction processor:

```
      # Assume that R0 is permanenlty set to 0
      MOV   R1, #1
      MOV   R2, #0
      MOV   R7, #7
loop: MOV   R3, R2
      ADD   R4, R3, R9
      SUB   R5, R4, R7
      JUMPZ R5, done
      ADD   R2, R2, R1
      JUMPZ R0, loop
done:
```

   (a) **(10 Points)** Write out the equivalent C code to the above sequence of assembly code. You may choose arbitrary variable names for the registers used.

   (b) **(20 Points)** For the first 6 instructions, list the basic register/memory transfer operations that occur during each clock cycle. Also for each clock cycle, provide the contents of the registers/memory locations whose contents have changed.

   An example of how to get started in included below:

```
Cycle       Events and RF/Mem Contents
--------------------------------------
0000        IR = I[0], PC = PC + 1
0001        IR=0x31X1, PC=1
0002        RF[1] = 1 (RTL operation)
--------------------------------------
0003        IR = I[1], PC = PC + 1
            RF[1] = 1 (content changed)
```

4. **(20 Points)**

   Using the six-instruction processor ISA (and if desired, an unconditional JMP instruction), write an assembly program for the following C code, which computes the sum of odd numbers less than N, where N is another name for $D[9]$ and N is an odd number. *Hint*: Use a register to first store N.

```
int i = 1;
int sum = 0;
while (i != N) {
    sum = sum + i;
    i = i + 2;
}
```