ECE 375
Computer Organization and Assembly Language Programming
Winter 2015
Assignment #1

[25 pts]

1- Consider a 1-address CPU that has a memory unit with 128K words of 32 bits each. An instruction is stored in one memory word. The instruction format is divided into four fields: opcode field, 2-bit addressing mode field that specify direct, indirect, indirect with pre-decrement, or indirect with post-increment addressing mode, a register field that specifies one of 32 registers, and an address field. For your information, given an *address*
   - Direct addressing is where the operand is located in M[*address*].
   - Indirect addressing is where the operand is located in M[M[*address*]].
   - Indirect addressing with pre-decrement is where the operand is located in M[-M[*address*]].
   - Indirect addressing with post-increment is where the operand is located in M[M[*address*]+].

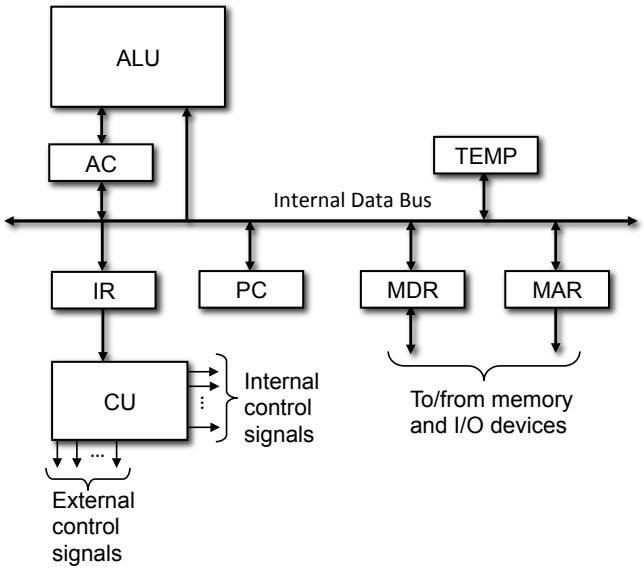
   (a) What is the maximum number of opcodes that can be incorporated into the CPU? How many bits are in the opcode field, the register field, and the address field? Draw the instruction format and indicate the number of bits in each field.
   (b) How many bits are required for the registers PC, MAR, MDR, IR, and AC?

[25 pts]

2- Consider the following hypothetical 1-address assembly instruction called "Store Accumulator with Post-increment" of the form

   STA (x)+ ; M(M(x)) ← AC, M(x) ← M(x)+1

   Suppose we want to implement this instruction on the pseudo-CPU discussed in class augmented with a temporary register TEMP. An instruction consists of 16 bits: A 4-bit opcode and a 12-bit address. All operands are 16 bits. PC and MAR each contain 12 bits. AC, MDR, and TEMP each contain 16 bits, and IR is 4 bits. Give the sequence of *microoperations* required to implement the Execute cycle (Fetch cycle is given below) for the above STA (x)+ instruction. Your solution should result in exactly 8 microoperations. Assume PC is currently pointing to the STA (x)+ instruction and only PC and AC have the capability to increment/decrement itself.



Fetch Cycle

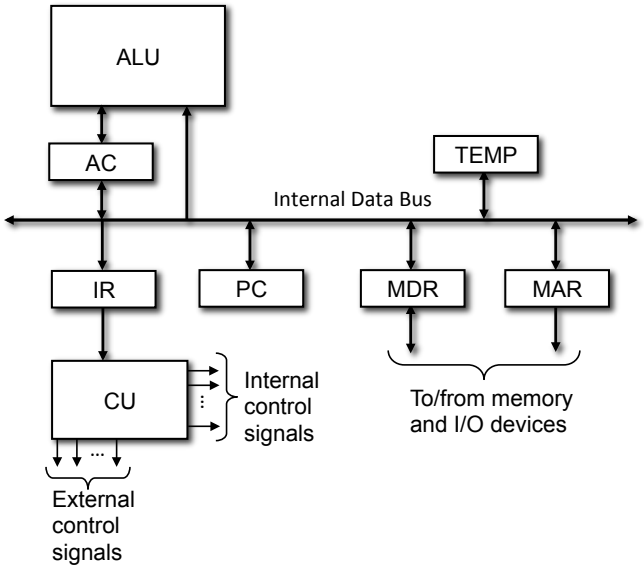
Step 1: MAR ← PC;
Step 2: MDR ← M(MAR), PC ← PC+1 ; Read inst. & increment PC
Step 3: IR ← MDR_{opcode}, MAR← $MDR_{address}$

[25 pts]

3- The PDP-8 was the first successful commercial minicomputer, produced by Digital Equipment Corporation (DEC) in the 1960s (see http://en.wikipedia.org/wiki/PDP-8). One of the assembly instructions it supported was "Increment and Skip if Zero (ISZ)" of the form

ISZ Y ; M(Y) ← M(Y) + 1, If(M(Y)+1=0) Then PC ← PC +1

Suppose the pseudo-CPU discussed in class with a temporary register (TEMP) shown below can be used to implement the ISZ instruction. Give the sequence of *microoperations* required to fetch and execute ISZ. Your solution should result in minimum number of microoperations. You must not destroy the original content of the AC. Assume PC is currently pointing to the instruction and only PC and AC have the capability to increment itself.



[25 pts]

4- Based on the initial register and data memory contents shown below (represented in hexadecimal), show how these contents are modified (in *hexadecimal*) after executing each of the following AVR assembly instructions. Do not be concerned about what happens to the Status Register (SREG) *after* the operation. *Instructions are unrelated*.

(i) `CPI R28, 2`
(ii) `STD Y+0x05,R4`
(iii) `LD R3, X+`
(iv) `EOR R1, R3`
(v) `ADIW R27:R26, 8`