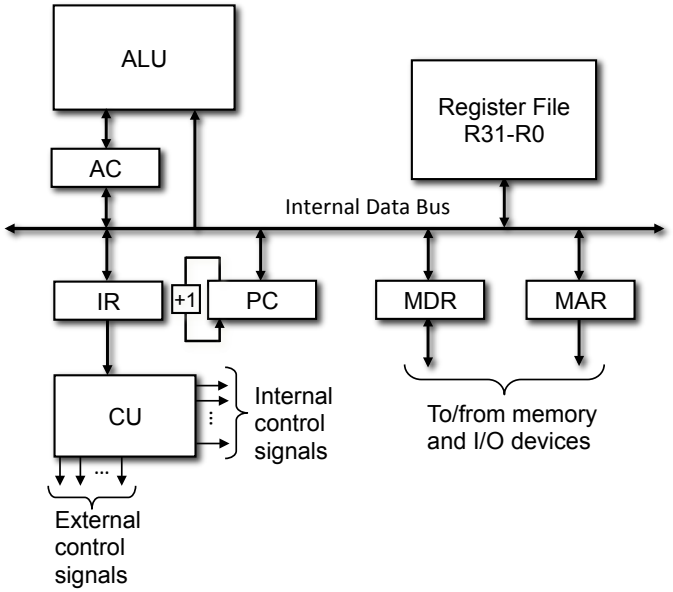
ECE 375
Computer Organization and Assembly Language Programming
Winter 2015
Assignment #2

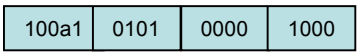
[25 pts]

1- Consider the internal structure of the pseudo-CPU discussed in class augmented with a *single-port register file* (i.e., only one register value can be read at a time) containing 32 8-bit registers (R0-R31). Suppose the pseudo-CPU can be used to implement the AVR instruction `ST -X, R3`. Give the sequence of microoperations required to Fetch and Execute AVR's `ST -X, R3` instruction. *Your solutions should result in exactly 6 cycles for the fetch cycle and 7 cycles for the execute cycle.* You may assume only the AC and PC registers have the capability to increment/decrement itself. Assume MDR register is 8-bit wide (which implies that memory is organized into consecutive addressable bytes), and AC, PC, IR, and MAR are 16-bit wide. Also, assume Internal Data Bus is 16-bit wide and thus can handle 8-bit or 16-bit (as well as portion of 8-bit or 16-bit) transfers in one microoperation. Clearly state any other assumptions made.

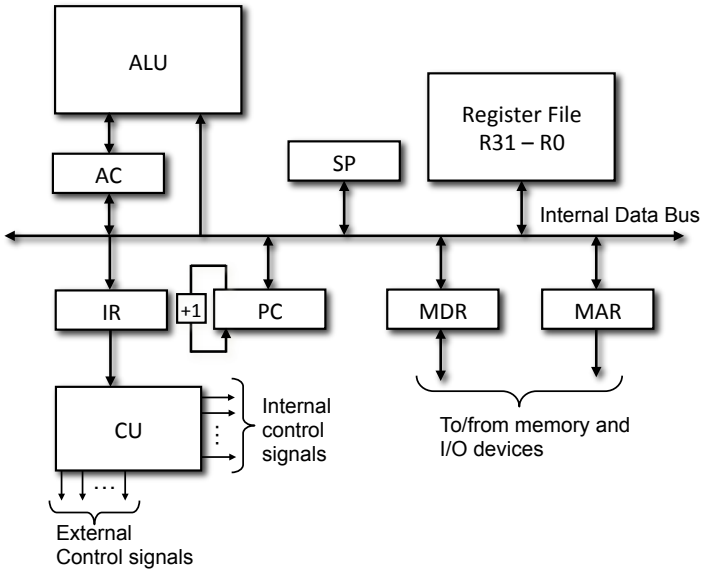


[25 pts]

2- Consider the internal structure of the pseudo-CPU discussed in class augmented with with a *single-port register file* (i.e., only one register value can be read at a time) containing 32 8-bit registers (R0-R31) and a Stack Pointer (SP) register. Suppose the very simple CPU can be used to implement the AVR instruction `RET` (Return from Subroutine) with the format shown below:



`RET` pops the return address from the stack and jumps to the return address. Give the sequence of microoperations required to Fetch and Execute AVR's `RET` instruction. *Your solutions should result in minimum number of microoperations*. Assume the memory is organized into addressable bytes (i.e., each memory word is a byte), MDR register is 8-bit wide, and SP, PC, IR, and MAR are 16-bit wide. Also, assume Internal Data Bus is 16-bit wide and thus can handle 8-bit or 16-bit (as well as portion of 8-bit or 16-bit) transfers in one microoperation and SP has the capability to increment itself. *Clearly state any other assumptions made*.



[25 pts]

3- Using AVR assembly language, write a subroutine XOR that evaluates the logical exclusive-OR of two operands *A* and *B*. The subroutine will be implemented using a skeleton code shown below:

```
.ORG    0x000F
        RCALL  XOR
        …
           …
.ORG    0X100F
XOR:    …   ; Your code goes here
        …   ;
        RET
```

(a) Assume the value *A* is in R1 and value *B* in R2 when the subroutine is called. However, there is a catch! You can use any AVR assembly instructions **except** `EOR` and `AND` instructions. In addition, you may not destroy (overwrite) the original contents of registers R1 and R2.

(b) Show the contents of the stack right after `RCALL` is made.

[25 pts]

4- Determine the location (i.e., address) and binary machine code for each instruction in the code developed for Problem #3. Examples of addresses and machines codes for `RCALL` and `RET` are shown below.

```
                                Address          Binary
        .ORG    0x000F
        RCALL   XOR             000F:     1101   kkkk   kkkk   kkkk
        …                       …
        .ORG    0x0046
XOR:    MOV     R3, R1          0046:     0010   11rd   dddd   rrrr
        COM     R3              0047:     1001   010d   dddd   0000
        OR      R3, R2          0048:     0010   10rd   dddd   rrrr
        COM     R3              0049:     1001   010d   dddd   0000
        MOV     R4, R2          004A:     0010   11rd   dddd   rrrr
        COM     R4              004B:     1001   010d   dddd   0000
        OR      R4, R1          004C:     0010   10rd   dddd   rrrr
SKIP:   COM     R4              004D:     1001   010d   dddd   0000
        OR      R3, R4          004E:     0010   10rd   dddd   rrrr
        RET                     004F:     1001   0101   0000   1000
```