# CS1S Computer Systems: Questions and Solutions
# May 2016

**Duration: 1 hour**
**Rubric: Answer both questions**
**This examination is worth a total of 50 marks**

**1.** **(a)** Convert 1110 1011 to a decimal number, assuming binary representation.

[2]

128 + 64 + 32 + 8 + 2 + 1= 235 by adding the powers of 2 corresponding to the positions where there is a 1 bit in the word. [Problem solving.]

**(b)** Convert 1110 1011 to a decimal number, assuming two's complement representation.

[3]

Since the leftmost bit is 1, this is a negative number. Negate it to get a nonnegative number. To negate, first invert giving 0001 0100. Then increment, giving 0001 0101. Now this is nonnegative so its binary representation is the same as its two's complement value; this is 16 + 4 + 1 = 21. Since the negation of the original word is 21, the answer is -21. [Problem solving. 1 mark for identifying it as negative; 2 marks for negation.]

**(c)** All of the four-digit constants in this question are hexadecimal numbers. Suppose R1 = 002b; R2 = 0004; R3 = ffff. Also suppose mem[002b] = 00a3; mem[002c] = 0c21; mem[002d] = ca00. Consider the following Sigma16 instructions. Give the values of registers R4, R5 and R6 after the instructions are executed.

```
load   R4,$0002[R1]
lea    R5,$0003[R1]
add    R6,R2,R3
```

[3]

R4 = mem [0002 + 002b] = mem [002d] = ca00. R5 = 0003 + 002b = 002e. R6 = 4 + (-1) = 0003. [Basic problem, requires understanding addressing, memory accesses, and integers.]

**(d)** Explain what the Sigma16 instruction `trap R1,R2,R3` does. Give two differences between a `jump` instruction and a `trap` instruction, and explain why `trap` (rather than `jump`) is used to make a request to the operating system.

[5]

The instruction jumps to a fixed address. This address is determined by the hardware and is not specified by the instruction. That address should be within the operating system, and is the beginning of the code that handles exceptions. Thus the trap is a jump but to an implicit address rather than to an address specified in the instruction. In addition, the trap instruction changes the system status flag from User state to System state; this allows the processor to execute privileged instructions. The operating system may need to execute privileged instructions in order to perform the action requested by the user. The register operands are used to allow the user to provide information to the operating system, including a trap code (specifying which request the user is making), and arguments (e.g. the address of a buffer for I/O). There are two major reasons for using trap rather than jump to request an operation. First, the system status must be changed from User to System in order to enable the operating system to execute the necessary instructions. Second, the user program does not need to know the address of the trap handler; this could be changed from one operating system or machine to another, and on some architectures it can be specified by a register or firmware. In any of these cases, the use of a jump instruction would require reassembling the user program. Another point of style is that it violates separation of concerns to require the user program to know addresses internal to the operating system. [Bookwork and synthesis. 2 marks for explanation of instruction, 1 mark for each difference, 1 marks for reason.]

**(e)** There is an array named `x` that contains `n` integers, where `n` is an integer variable in memory. Write a Sigma16 assembly language program that overwrites each element of the array with its absolute value. The program should define `n = 6` and define the initial elements of the array as 3, 0, -41, 7, -9, 2. After the program runs the elements should be 3, 0, 41, 7, 9, 2, but the program must work correctly for any nonnegative `n` and initial array elements.

[12]

```
; R1 = i = loop index
; R2 = n = array size
; R3 = constant 1
; R4 = y = x[i]
      add    R1,R0,R0     ; i= 0
      load   R2,n[R0]     ; R2 = n
      lea    R3,1[R0]     ; R4 = 1
loop  cmp    R1,R2        ; i <=> n
      jmpge  done[R0]     ; if i >= n then goto done
      load   R4,x[R1]     ; y = x[i]
      cmp    R4,R0        ; i <=> 0
      jmpge  skip[R0]     ; if i >= 0 then goto skip
      sub    R4,R0,R4     ; y = 0-y
skip  store  R4,x[R1]     ; x[i] = y
      add    R1,R1,R3     ; i = i + 1
      jump   loop[R0]     ; goto loop
done  trap   R0,R0,R0     ; terminate
n     data   6
x     data   3
      data   0
      data -41
      data   7
      data  -9
      data   2
```

[Problem solving, requires understanding of the instruction set, conditionals, indexed addressing and loops.  3 marks for initialization, 3 marks for loop, 3 marks for array access and 3 marks for absolute value.]

**2.** **(a)** Describe the behavior of a 1-bit register (the reg1 circuit). Draw a diagram of the circuit.

[4]

The register has two input bits load and x, produces an output r, and contains a 1-bit state. The output r is always set to the state; this is the register readout. The state remains constant during a clock cycle, and may change only at a clock tick. At a tick, if load=1 then the old state is discarded and replaced with the value of x. If load=0 then the state remains unchanged, and x is ignored. The circuit is reg1 load x = r where r = dff (mux1 load r x). (That is a textual Hydra specification, but the students will use the corresponding diagram rather than the hardware description language.) [Bookwork. 2 marks for behavior, 2 marks for implementation.]

**(b)** Explain the purpose of the clock in a synchronous circuit. Describe how a suitable clock speed for the circuit is determined.

[5]

The clock generates a sequence of ticks, which define points in time. The clock signal is connected to every flip flop, and a flip flop updates its state only at the tick, so the effect is to cause all flip flops to update their states simultaneously. Without this, it could happen that one flip flop updates before another, and its output changes leading to a spurious change of the input to the second flip flop. By synchronizing all flip flops, the entire machine has a single state which is a vector of all the individual flip flop states. The clock must run slowly enough to allow all combinational signals to settle down to a stable final value before the next tick. This means the cycle must be long enough for the slowest combinational path to settle down; this is the critical path. In addition, some additional time is added to the clock period to allow for random variations in component speeds. [Bookwork plus synthesis. 3 marks for clock and flip flops, 2 marks for speed.]

**(c)** Give two reasons that user programs are not allowed to execute Input/Output instructions directly, but must request the Operating System to perform I/O.

[4]

(1) It would be difficult or impossible for a user to coordinate its I/O with that of other programs. It would not know which I/O devices it can use, and a simple action like writing text to an output file would be complicated. (2) It would compromise security; a user program could overwrite files belonging to other users, or read private data. (3) I/O is device dependent and often quite complicated; it is undesirable anyway for a user program to do it as the code would be complex and would need to be changed every time an I/O device is updated. [Bookwork plus synthesis. Other reasons are possible as well; 2 marks for each reason.]

**(d)** Define the term *privileged instruction*. Explain how privileged instructions prevent the user from performing I/O directly.

[4]

A privileged instruction can be executed by the operating system but not by the user program. To enforce this, there is a control flag in the hardware that determines whether the processor is executing in User state or System state. If a privileged instruction is executed and the machine is in System state, the instruction is executed, but if it is in User state the control performs an interrupt. This sets the processor to System state and transfers control to the operating system, which can decide what to do (e.g. abort the user program). The purpose of this is to prevent the user program from executing instructions that could violate system security; these instructions are privileged. I/O instructions are privileged, so the user program must request the operating system to do I/O for it, rather than doing the I/O itself. [Bookwork plus synthesis. 2 marks for definition, 2 marks for the effect.]

**(e)** Host A is sending packets through the Internet to Host B, and these packets normally go through a router R. Suppose the router R fails because it loses power. Explain how the Internet can eventually recover and regain the ability to send packets from A to B.

[8]

Each router periodically sends routing protocol (RIP) messages to its neighbors, and receives such messages from its neighbors. These messages are used to update the routing tables. When router R fails, its neighbors will eventually fail to receive expected messages from R, and will update their tables to indicate that R isn't there. After that, when a packet arrives it will be sent on a different path. Even if the neighbor of R no longer has connectivity to B, this information will spread back and is likely eventually to reach a router that has an alternative route to B. Subsequent packets from A will then take this alternative route. A number of packets may be lost, but eventually connectivity will be reestablished. In this way, the Internet continually adjusts to changes in the network topology. [Bookwork plus synthesis. 4 marks for RIP packets, 4 marks for adjusting the routing tables.]