CS 061 – Computer Organization **Fall - 2013**

Quiz 9 – Thursday 12/5 solution

1. In a processor that uses the technique of memory mapping to address ports *(registers that interface between the cpu and peripherals)*, how are the ports actually accessed *(i.e. written to/read from)*?
   - **a.** via dedicated i/o instructions.
   - **b.** via interrupts
   - **c.** via polling
   - **d. via standard Load/Store instructions**
   - **e.** via a status register

2. What component of the cpu gets "interrupted" by an Interrupt signal? (i.e. where does the Interrupt signal go to?)
   - **a.** The PC
   - **b.** The PC-MUX
   - **c.** The MAR-MUX
   - **d.** The global bus
   - **e. The FSM**
   - **f.** The Memory Mapping logic

3. When an Interrupt is accepted, current state information is saved by pushing it on a stack. Why is a stack used rather than simply saving it to a register or a fixed location in RAM?
   - **a.** Multiple memory reads are very time-consuming: pushing to a stack is quicker
   - **b. To allow for the possibility of nested Interrupts**
   - **c.** To allow the cpu to differentiate between routines with Supervisor privilege and User privilege
   - **d.** Saving the state information to memory would interfere with the process of loading the Interrupt Service Routine from memory
   - **e.** There are no major differences between the two approaches – it is just a minor design decision.

4. In the LC-3, the TRAP instruction and the interrupt handler both manage the invocation of service routines in a similar fashion. Specifically, both use:
   - **a.** polling of status registers to decide when to read from/write to a port
   - **b. a trap/interrupt vector as an entry point into a table of service routine addresses**
   - **c.** an IACK signal to determine which service routine is requested
   - **d.** a stack to store information required for the return, allowing nested calls
   - **e.** a system of task priority comparisons to determine whether to invoke the service routine

5. What is the purpose of the IACK signal in processing an interrupt?:
   - **a.** It forces the CPU to alter the control sequence to acknowledge the interrupt
   - **b. It interrogates the peripheral devices to find out which set the interrupt**
   - **c.** It contains the interrupt vector
   - **d.** It is the response of the peripheral device after the interrupt is acknowledged
   - **e.** None of the above

6. What system state information has to be saved before an interrupt-enabled LC-3 can proceed with servicing an interrupt?
   - **a.** the value of every control signal produced by the finite state machine
   - **b.** the value of every control signal produced by the finite state machine, plus the contents of all Registers (GPRs, PC, condition codes, etc.) – except the IR
   - **c.** the PC
   - **d.** the PC and all the General Purpose Registers
   - **e. the PC, and the PSR (= the NZP condition codes, the Privilege level, and the current task priority)**
   - **f.** the PC and the MCR (Machine Control Register)

7. The two main approaches to converting a HLL (Higher Level Language) source code to ML (Machine Language) are:
   a. direct and indirect
   b. memory mapping and polling
   c. assembly and disassembly
   d. **interpreting and compiling**
   e. compiling and linking

8. The structure which allows Higher Level Languages to make nested function calls is:
   a. **Activation records stored on the run-time stack**
   b. The symbol table
   c. The frame pointer
   d. The Processor Status Register
   e. The Machine Control Register

9. Which LC-3 assembly language instruction is most likely to be used to compile a HLL (Higher Level Language) access to a local variable:
   a. **LDR**
   b. LDI
   c. LD
   d. TRAP
   e. JSR

10. Which register is most likely to be used as a pointer in accessing the variable, in the situation described in the previous question?
   a. **Frame pointer (R5 in the LC3)**
   b. Top of Stack Pointer (R6 in the LC3)
   c. Program Counter (R7 in the LC3)
   d. Processor Status Register
   e. Machine Control Register