CS 2200 - Introduction to Systems

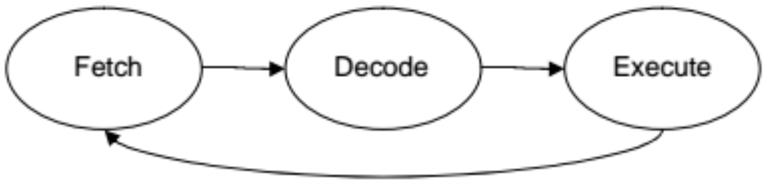
Fall 2016

Homework 4

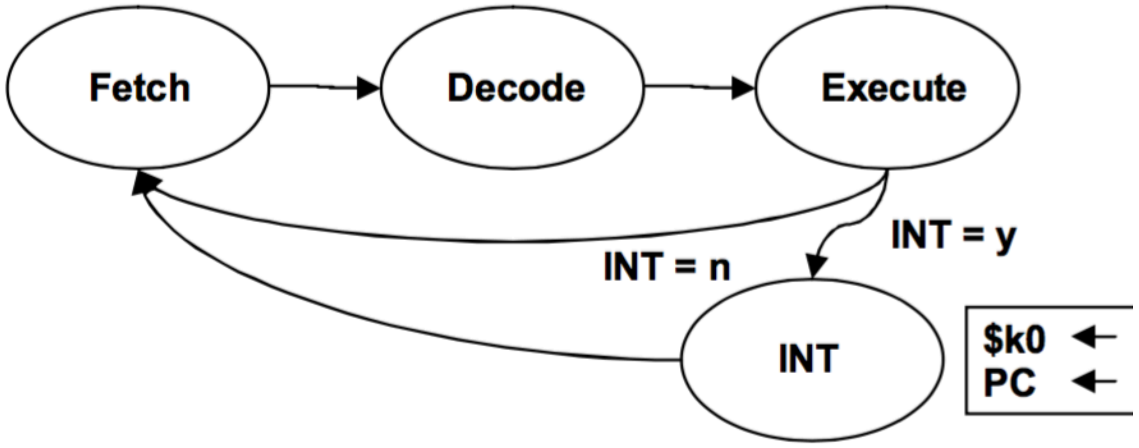
Rules:

- Please print a copy of the assignment and handwrite your answers. No electronic submissions are allowed. **Please print as one double-sided page.**
- This is an individual assignment. No collaboration is permitted.
- Due Date: **28st September 2016** – 6:05 PM (Start of recitation). Bring your BuzzCards.

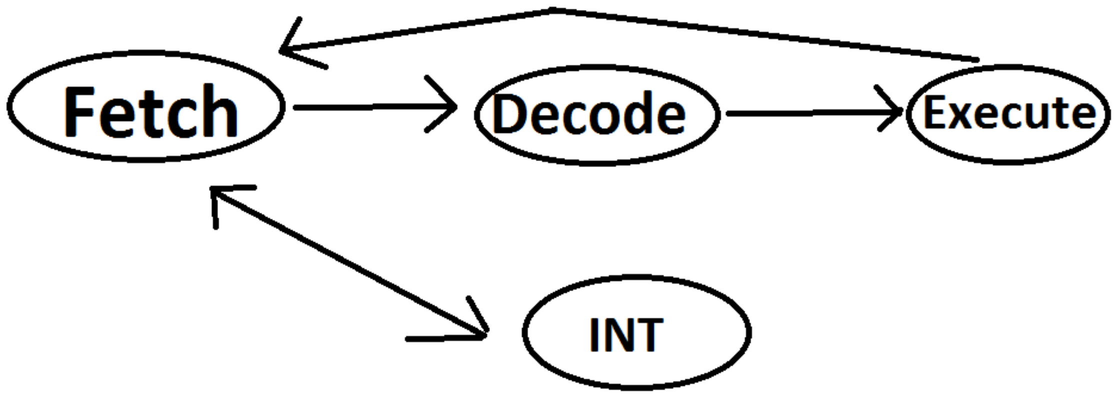
Name (please print):________________________ GTLogin:_________________ Section____



1. Above is the finite state machine for a processor that does not support interrupts. In the space below, draw the updated finite state machine for a processor that supports interrupts. **[10 Pts]**



OR



2. What happens in hardware when the LC 2200 processor receives an interrupt. Write down the **4 essential** steps the processor takes before it can execute the interrupt handler code. **[20 Pts]**

1. Save PC to $k0. Why? Program needs to know where to jump back to after handling the interrupt.
2. Acknowledge the interrupt with the INTA line and get the device vector on the bus.
3. Use the device vector to index into IVT to get the handler address and store it in PC
4. Disable Interrupts. Why? Support for nested interrupts.
5. Extra: save old mode bit and set mode bit to kernel

3. What does the RETI instruction do? **Explain why** it is necessary to make RETI atomic. **[15 Pts]**

RETI returns from the interrupt handler code. That includes putting $k0 into the PC, enabling interrupts, and restoring the mode bit. We need an atomic RETI instruction so that all of those tasks happen at one time and cannot be interrupted. If we used multiple instructions to accomplish those tasks, we would have to enable interrupts BEFORE jumping back to the value in $k0. (Obviously, if we jumped before enabling interrupts, we would never reach our instruction to enable the interrupts!) If we enable interrupts, we could be interrupted just before jumping. Then, what would happen to our $k0 value?! We would lose it and would be unable to return.

4. Sometimes during the course of program execution, more than one device can issue an interrupt at the same time. How does the CPU decide which interrupt to handle first? **Name and explain** 2 strategies used to tackle this situation. **[20 Pts]**

We need some way of specifying which device should have priority! So, we can:

- Daisy Chain: Link the devices together in a line. The INTA line would first reach the device closest along the daisy chain, and if that device asserted the INT line, it would put its vector on the bus. If not, it would send the INTA signal along to the next device in the chain.
- Priority Encoder: A logic unit would take in input from all devices and decide which of the devices should be the one to put its vector on the bus.

5. Below is a snippet from an interrupt handler written in LC 2200 Assembly. However, it is missing some crucial lines of assembly code. Your task is to fill in the missing pieces. (**Hint:** Refer to your class notes and read the book to learn exactly what sequence of tasks the interrupt handler performs) **[35 Pts]**

Some helpful instructions that are a part of the LC2200 – Interrupt ISA:
ei – Enables interrupts
di – Disable interrupts
reti – Return Enable Interrupt

```
ti_inthandler:

        sw $k0, -1($sp)

        ei                                   ! Enable Interrupts
        sw $s0, -2($sp)

        sw $s1, -3($sp)

        la $s0, counter                      !incrementing counter
        lw $s0, 0($s0)                       !s0 = 0xFFFFFC
        lw $s1, 0($s0)                       !s1 = contents of 0xFFFFFC
        addi $s1, $s1, 1
        sw $s1, 0($s0)
        lw $s1, -3($sp)
        lw $s0, -2($sp)
        di
        lw $k0, -1($sp)
        reti

Counter: 0xFFFFFC
```