Computer Architecture IV

AGENDA

- Guided Project Pt. III
- Subroutines/Function Calls
- Interrupts
- *I will hold an AMA tomorrow 6-7 PM PST*

GUIDED PROJECT PT. III

- Implement the system stack and be able to run the *stack.ls8* program

Subroutines/Function Calls

SUBROUTINES/FUNCTION CALLS

- To support functions in programming languages, we need to be able to tell the computer to jump to a certain part in memory and execute instructions
- Once those instructions are done, we need to be able to jump back to where we were originally and keep executing

SUBROUTINES/FUNCTION CALLS

- The computer implements subroutines by using the **CALL** and **RET** instructions
- **CALL** takes in a register and jumps to the address stored in that register. It also saves the address of the next instruction onto the stack so it knows where to jump back after the subroutine is done.
- **RET** doesn't take any operands and simply pops the topmost element off the stack and sets the PC to its value

CALL

- **CALL** takes in a register and jumps to the address stored in that register. It also saves the address of the next instruction onto the stack before jumping to that address.
- CALL 0x3 *Jump to the address stored in register 3*
  - Store the address of the next instruction onto the stack
  - Set the PC to the address stored in register 3

RET

- **RET** doesn’t take any operands and simply pops the topmost element off the stack and sets the PC to its value

Subroutines/Function Calls Demo

INTERRUPTS

- A way for the CPU to handle external events (e.g. typing on a keyboard, moving the mouse, timers, etc.) while saving the current state of execution
- Interrupts can also be called internally via the **INT** instruction
- During an interrupt, the CPU jumps to execute the subroutine/instructions provided by the interrupt handler

INTERRUPTS

- A CPU uses a couple of registers to determine which interrupts are enabled and are to be executed
    - **IM (Interrupt Mask)** - controls which interrupts should be handled
    - **IS (Interrupt Status)** - keeps track of which interrupts have been issued
- A CPU has an interrupt vector table, containing the address of the interrupt subroutine

INTERRUPTS

- During an interrupt
  - Values in registers are stored onto the stack
  - The address of the instruction to jump to is looked up in the interrupt vector table
  - The PC is set to that address
- An interrupt finishes with the IRET command
  - Register values are popped off from the stack
  - The PC is set to the correct instruction
- See *interrupts section* in spec for more details

INTERRUPTS

- For your stretch goal, implement the timer and keyboard interrupts
- *Warning: This is not that trivial*