Computer Architecture
(CSC-3501)
Lecture 12
(26 Feb 2008)

Seung-Jong Park (Jay)
http://www.csc.lsu.edu/~sjpark

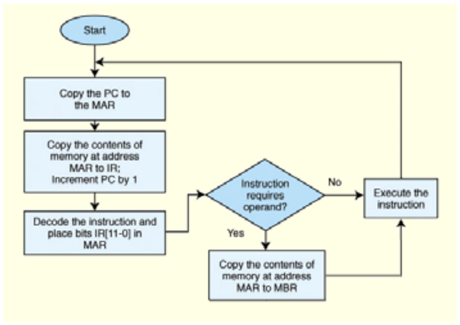
Announcement

4.9 Instruction Processing

- The *fetch-decode-execute cycle* is the series of steps that a computer carries out when it runs a program.
- We first have to *fetch* an instruction from memory, an d place it into the IR.
- Once in the IR, it is *decoded* to determine what need s to be done next.
- If a memory value (operand) is involved in the operati on, it is retrieved and placed into the MBR.
- With everything in place, the instruction is *executed*.

The next slide shows a flowchart of this proc ess.

4.9 Instruction Processing

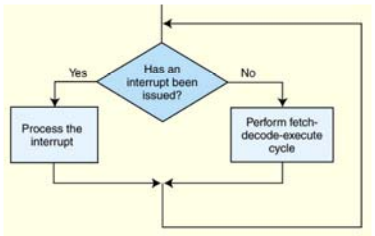

Start
Copy the PC to the MAR
Copy the contents of memory at address MAR to IR; Increment PC by 1
Decode the instruction and place bits IR[11-0] in MAR
Instruction requires operand?
No
Yes
Execute the instruction
Copy the contents of memory at address MAR to MBR

4.9 Instruction Processing

- All computers provide a way of interrupting the fet ch-decode-execute cycle.
- Interrupts occur when:
  - A user break (e.,g., Control+C) is issued
  - I/O is requested by the user or a program
  - A critical error occurs
- Interrupts can be caused by hardware or software .
  - Software interrupts are also called *traps*.

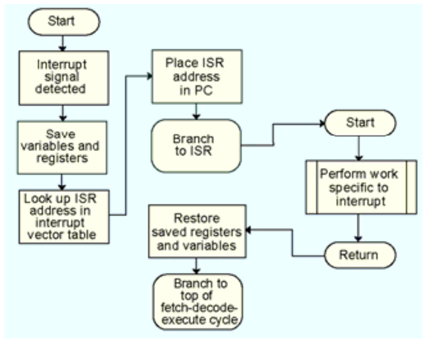
4.9 Instruction Processing

- Interrupt processing involves adding another step to t he fetch-decode-execute cycle as shown below.


Has an interrupt been issued?
Yes
No
Process the interrupt
Perform fetch-decode-execute cycle

The next slide shows a flowchart of “Process the interr upt.”

4.9 Instruction Processing


Start
Interrupt signal detected
Save variables and registers
Look up ISR address in interrupt vector table
Place ISR address in PC
Branch to ISR
Start
Perform work specific to interrupt
Return
Restore saved registers and variables
Branch to top of fetch-decode-execute cycle

4.9 Instruction Processing

- For general-purpose systems, it is common to dis able all interrupts during the time in which an inter rupt is being processed.
  - Typically, this is achieved by setting a bit in the flag s register.
- Interrupts that are ignored in this case are called *maskable*.
- *Nonmaskable* interrupts are those interrupts that must be processed in order to keep the system in a stable condition.

4.9 Instruction Processing

- Interrupts are very useful in processing I/O.
- However, interrupt-driven I/O is complicated, and is beyond the scope of our present discussion.
  - We will look into this idea in greater detail in Chapt er 7.
- MARIE, being the simplest of simple systems, us es a modified form of programmed I/O.
- All output is placed in an output register, OutREG, and the CPU polls the input register, InREG, until input is sensed, at which time the value is copied i nto the accumulator.