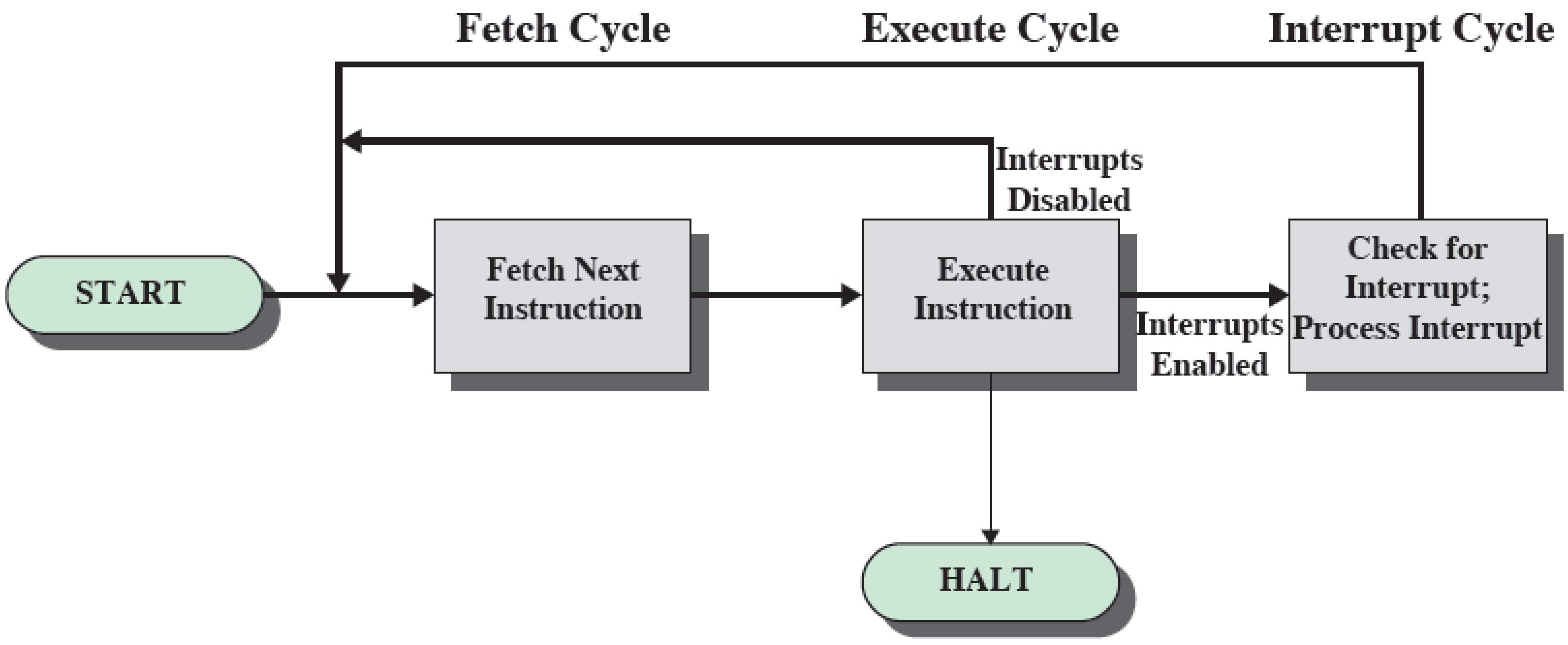
CS430 Computer Architecture

Spring 2015

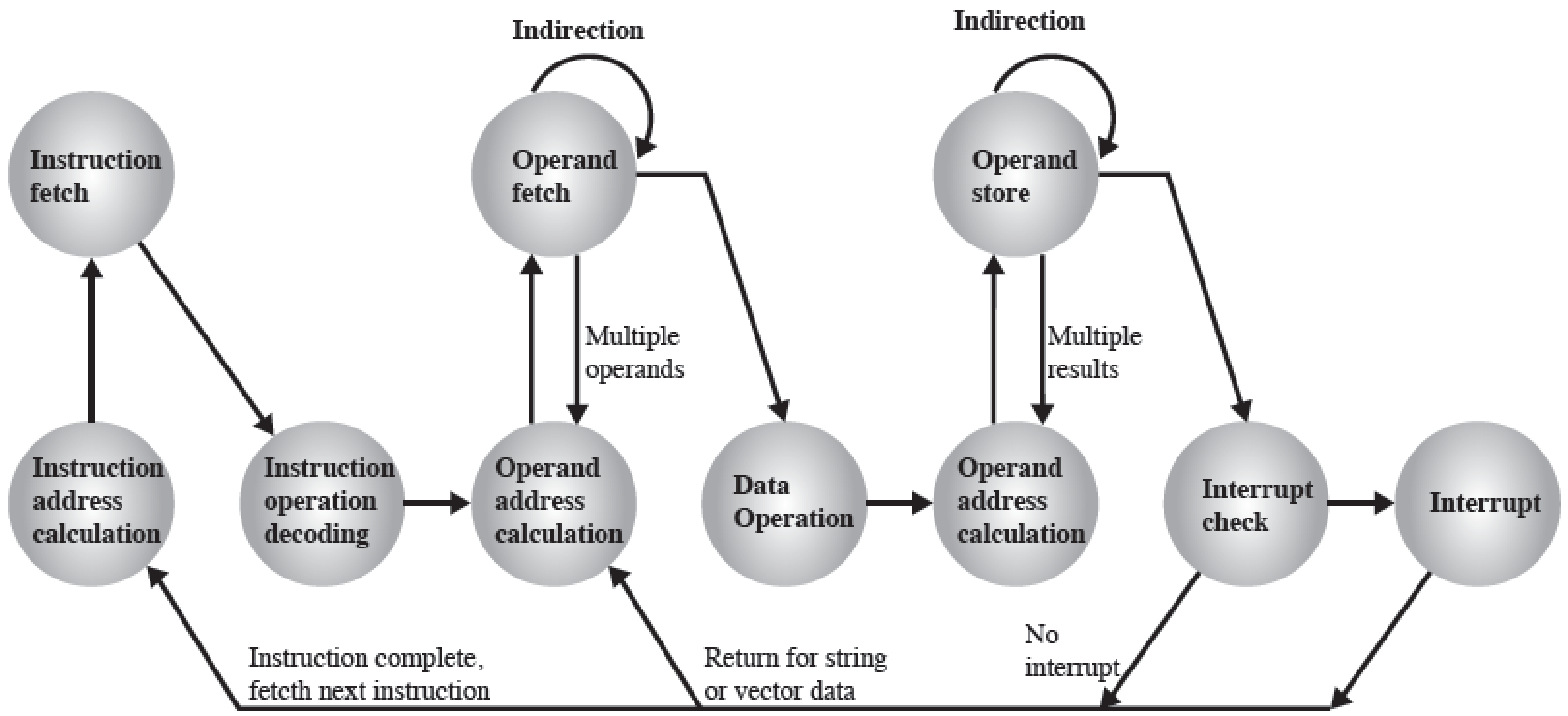
Chapter 14
Processor Structure and Function

- Instruction Cycle from Chapter 3



Updated Instruction Cycle

- Give an x86 instruction that goes through as many stages as possible


Indirection
Indirection
Instruction fetch
Operand fetch
Operand store
Multiple operands
Multiple results
Instruction address calculation
Instruction operation decoding
Operand address calculation
Data Operation
Operand address calculation
Interrupt check
Interrupt
Instruction complete, fetcth next instruction
Return for string or vector data
No interrupt

Instruction Pipelining

- Organization enhancements to the processor over time have greatly improved performance.
- Examples include:
  - multiple general purpose registers versus a single accumulator
  - multiple levels of cache
  - next is pipelining

Instruction Pipelining

- Pipelining is a technique that improves instruction throughput by executing as many instructions as possible in parallel.
- In particular, these instructions are in various "stages" of pipeline execution.
- Instruction pipelining is very much like the assembly line at a manufacturing plant where products in various stages of production can be worked on simultaneously.

Instruction Pipelining

- We will break up instruction processing into the following six stages:
  1. FI (fetch instruction) - get the next instruction
  2. DI (decode instruction) - decode the opcode and operands
  3. CO (calculate operands) - calculate EA of the operands
  4. FO (fetch operands) - fetch operands from memory (not necessary for register data)
  5. EI (execute instruction) - execute instruction storing result if necessary
  6. WO (write operand) - write the result in MEM

Instruction Pipelining

- In a perfect world, this would lead us to an architecture that has a CPU with a six stage pipeline where each stage takes an equal duration of time to execute.

1. Do the above pipeline stages take the same duration of time to complete their specific operation? If not, give two stages that can have different durations of time to complete their task.
2. Assuming no pipelining and each stage in the pipeline takes 1 unit of time to complete, how long would it take 5 instructions to complete?

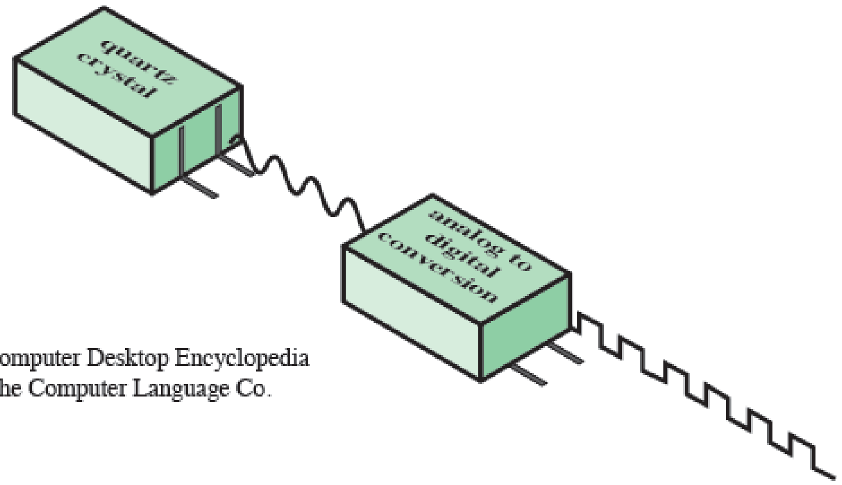
Instruction Pipelining

- Assuming a six stage pipeline and each stage in the pipeline takes 1 unit of time to complete, how long will it take 5 instructions to complete (assuming no delays)?
- Draw the timing diagram

Performance Assessment

- We begin taking a look at some traditional measures of processor speed
- Then we look at assessing processor and computer system performance

Clock Speed



From Computer Desktop Encyclopedia
1998, The Computer Language Co.

- Clock Rate (or Clock Speed) – the rate of pulses
- Clock Cycle (or Clock Tick) – a single clock pulse
- Clock Cycle Time – the time between ticks

Instruction Execution Rate

- A processor is driven by a clock with a constant frequency, f, or equivalently, a constant cycle time τ, where $\tau = \frac{1}{f}$
- I_c - instruction count is the number of machine instructions executed for a given program
- CPI – cycles per instruction is the **average** cycles taken to execute an instruction

Instruction Execution Rate

- Since each instruction type does not necessarily take the same number of cycles, CPI, is calculated as follows:
  - $CPI = \frac{\sum_{i=1}^{n}(CPI_i \cdot I_i)}{I_c}$ where
    - CPI_i is the number of cycles required for instruction type i
    - I_i is the number of executed instructions of type i
    - I_c is the total number of instructions executed
- The processor time, T, needed to execute a given program is $T = I_c \cdot CPI \cdot \tau$

MIPS Rate

- The MIPS rate is the measurement expressing millions of instructions per second
  - $MIPS\ Rate = \frac{f}{CPI \cdot 10^6}$
- Problem: A program results in executing 2 million instructions on a 1GHz processor. Given the following, compute the MIPS rate

Instruction Type	CPI	Instruction Mix (%)
Arithmetic & Logic	1	60
Load/Store w/cache hit	2	18
Branch	4	12
MEM reference w/cache miss	8	10

Speedup

- "Speedup" is a very simple, useful term that allows us a way to measure performance and execution times.
- $Speedup = Time(Old)/Time(New)$
- Example: A trip to downtown Portland takes 50 minutes by car and 30 minutes by MAX. What is the speedup? Answer: speedup=50/30=1.6666

Speedup

- New is faster than Old if Time(New) < Time(Old)
- New is N% faster than Old given $Speedup = \frac{Time(Old)}{Time(New)} = 1 + \text{N\%}$
- So, speedup of MAX over car is 1.6666; therefore, MAX is 66.66 faster than the car
- Notice also that $Speedup = Speed(New)/Speed(Old)$

Amdahl’s Law

- Given a job, some fraction of that job may be enhanced.
- In particular, if we can make some enhancement to a particular machine that improves performance, speedup will tell us how much faster the machine is with the enhancement versus the machine without the enhancement.

Amdahl’s Law

- Let's assume it takes us 1 hour to drive to Portland. From Portland, it takes us 3 hours by train to get to Seattle. What is the speedup if we fly instead and it takes 1 hour to fly to Seattle?
- Answer: speedup = Time(Old)/Time(New) = 4/2 = 2.0
- How much faster is the trip if we fly to Seattle versus taking the train?
- Answer: Flying is 100% faster than taking the train.

Fraction Enhanced

- Fraction enhanced is the fraction of the "original" time that can be enhanced.
- What is our fraction enhanced for our Trip to Seattle?
- Answer: 3 hours can be enhanced, so fraction enhanced is 3/4=75%.

Fraction Enhanced

- Further examples to clarify the fraction enhanced include the following:
  1. Speeding up the floating-point calculations doesn't speed up the integer calculations
  2. If we are traveling to Eugene and speedup the highway driving somehow, that doesn't speedup the packing, walking to the car, ...
- An interesting observation is that if we do the fraction enhanced at infinite speed, this gives us the maximum speedup for that enhancement
- What is the speedup of the six stage pipeline machine versus the machine with no pipelining?