



CPU and Memory - Advanced

**Reading: Chapter 8
(Except VLIW and EPIC)
(lightly read 8.4)**



Reference

- SoC

en.wikipedia.org/wiki/System-on-a-chip



Current CPU Architectures

- Current CPU Architecture Designs
 - Traditional modern architectures
 - Advanced architectures mentioned in text have not performed well (e.g., Intel Itanium/Merced)
- Current Popular CPU Architectures
 - Intel x86 family (including AMD)
 - IBM Mainframe series
 - ARM (mobile devices)
 - Sparc

Why are there so few CPU architectures in wide use?



Traditional Modern Architectures

Problems with early CPU Architectures and solutions:

- Large number of specialized instructions were rarely used but added hardware complexity and slowed down other instructions
- Slow data memory accesses could be reduced by increasing the number of general purpose registers
- Using general registers to hold addresses could reduce the number of addressing modes and simplify architecture design
- Fixed-length, fixed format instruction words would allow instructions to be fetched and decoded independently and in parallel



System Clock

- Different CPU operations take different amounts of time to complete
- The clock provides a master control as to when each step in the instruction cycle takes place
- Clock rate is a (not so good) measure of performance
- Clock rate is related to heat generation issues
- Clock rate is usually measured in time (e.g., nanosecond) or frequency (1 GHz = billion cycles/sec)
- Chip clock rate set at end of manufacturing process
- Recently, CPU power comes less from clock rate improvements than architectural improvements

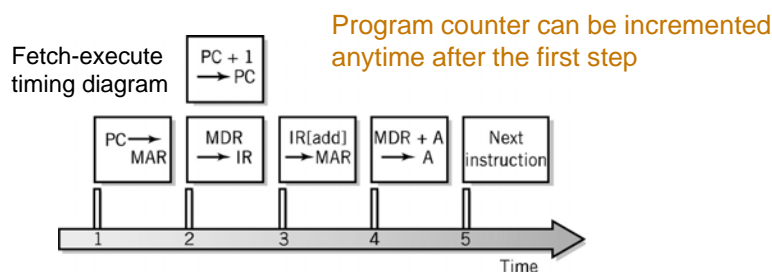
Copyright 2010-2011 John Wiley & Sons, Inc. & Robert F. Kelly

8-5



Fetch-Execute Cycle Timing Issues

- Computer clock is used for timing purposes for each step of the instruction cycle
- Instructions can (and often) take more than one step
- Data word width can require multiple steps



Copyright 2010-2011 John Wiley & Sons, Inc. & Robert F. Kelly

8-6



CPU Features and Enhancements

We will examine each of these features

- Separate Fetch/Execute Units
- Pipelining
- Multiple, Parallel Execution Units
- Scalar Processing
- Superscalar Processing
- Branch Instruction Processing

Copyright 2010-2011 John Wiley & Sons, Inc. & Robert F. Kelly

8-7



Separate Fetch-Execute Units

- Fetch Unit
 - Instruction fetch unit
 - Instruction decode unit
 - Determine opcode
 - Identify type of instruction and operands
 - Several instructions are fetched in parallel and held in a buffer until decoded and executed
 - IP – Instruction Pointer register holds instruction location of current instruction being processed
- Execute Unit
 - Receives instructions from the decode unit
 - Appropriate execution unit services the instruction

Copyright 2010-2011 John Wiley & Sons, Inc. & Robert F. Kelly

8-8



Instruction Pipelining

- Assembly-line technique to allow overlapping between fetch-execute cycles of sequences of instructions
- Scalar processing
 - Average instruction execution is approximately equal to the clock speed of the CPU
- Problems from stalling **Better performance with RISC**
 - Instructions have different numbers of steps
- Problems from branching

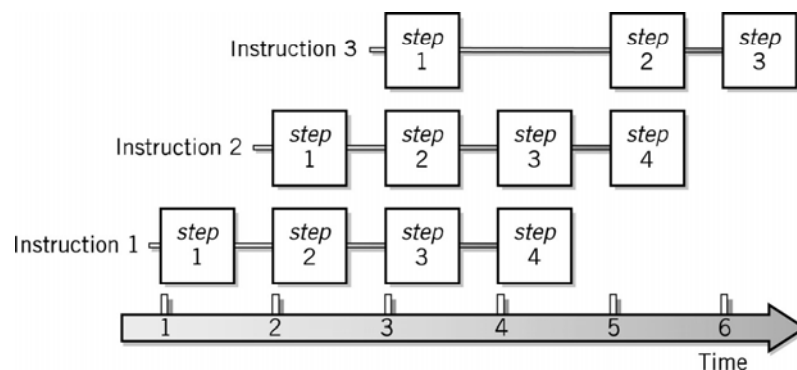
Some processors can execute close to one instruction per clock cycle – even though each instruction might take more than 4 clock cycles

Copyright 2010-2011 John Wiley & Sons, Inc. & Robert F. Kelly

8-9



Pipelining Example



Copyright 2010-2011 John Wiley & Sons, Inc. & Robert F. Kelly

8-10



Branch Problem Solutions

- Problem
 - pipeline contains multiple instructions, but a branch can change order of instructions
- Solutions
 - Separate pipelines for both possibilities
 - Speculative execution
 - Requiring the following instruction to not be dependent on the branch
 - Branch prediction

Copyright 2010-2011 John Wiley & Sons, Inc. & Robert F. Kelly

8-11



Multiple, Parallel Execution Units

- Different instructions have different numbers of steps in their cycle
- Differences in each step
- Each execution unit is optimized for one general type of instruction
- Multiple execution units permit simultaneous execution of several instructions

Typical instruction units are: load/store;
integer arithmetic, floating point
arithmetic, and branch

Copyright 2010-2011 John Wiley & Sons, Inc. & Robert F. Kelly

8-12



Superscalar Processing

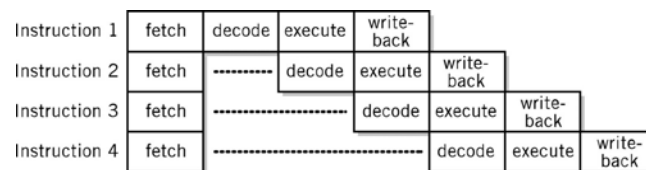
- Process more than one instruction per clock cycle, through
 - Pipelining
 - Multiple execution units
- Buffers for fetch and decode phases

Copyright 2010-2011 John Wiley & Sons, Inc. & Robert F. Kelly

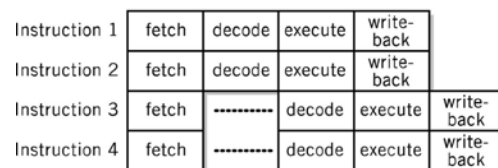
8-13



Scalar vs. Superscalar Processing



a. Scalar



b. Superscalar



Copyright 2010-2011 John Wiley & Sons, Inc. & Robert F. Kelly

8-14



Superscalar Issues

- Out-of-order processing – dependencies (hazards)
- Data dependencies
- Branch (flow) dependencies and speculative execution
- Parallel speculative execution or branch prediction
- Branch History Table
- Register access conflicts
 - Rename or logical registers

Copyright 2010-2011 John Wiley & Sons, Inc. & Robert F. Kelly

8-15



Memory Enhancements

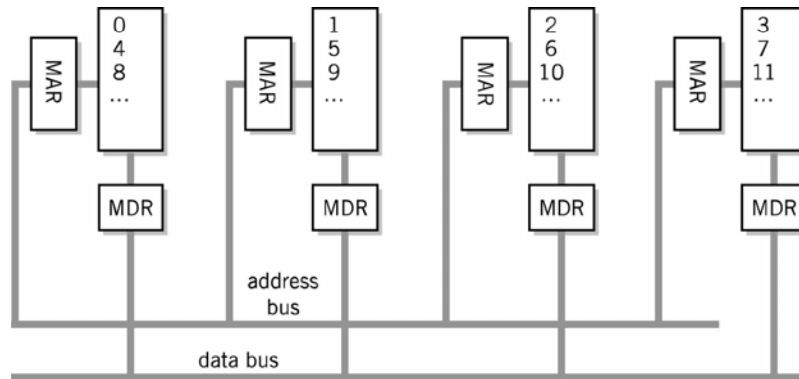
- Memory is slow compared to CPU processing speeds!
 - 2Ghz CPU = 1 cycle in $\frac{1}{2}$ of a billionth of a second
 - 70ns DRAM = 1 access in 70 millionth of a second
- Methods to improvement memory accesses
 - Wide Path Memory Access
 - Retrieve multiple bytes instead of 1 byte at a time
 - Memory Interleaving
 - Partition memory into subsections, each with its own address register and data register
 - Cache Memory

Copyright 2010-2011 John Wiley & Sons, Inc. & Robert F. Kelly

8-16



Memory Interleaving

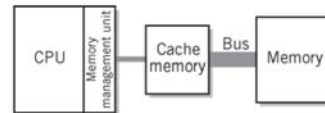


Copyright 2010-2011 John Wiley & Sons, Inc. & Robert F. Kelly

8-17



Cache Memory



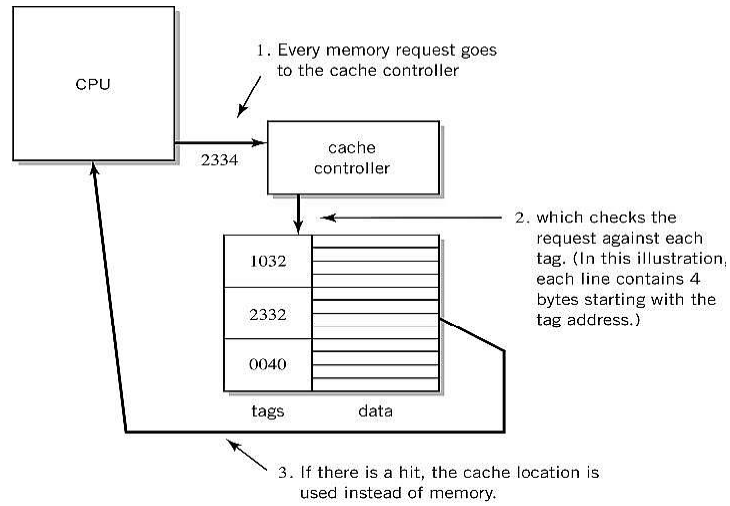
- High speed memory not directly accessible to the programmer that contains:
 - Blocks: referred to as a cache line
 - Tags: pointer to location in main memory
 - Cache controller - hardware that checks tags
- Hit Ratio: ratio of memory cache hits to total memory requests
- Synchronizing cache and memory
 - Write through – write to main memory whenever a cache line is change
 - Write back – write when cache line is replaced

Copyright 2010-2011 John Wiley & Sons, Inc. & Robert F. Kelly

8-18



Step-by-Step Use of Cache

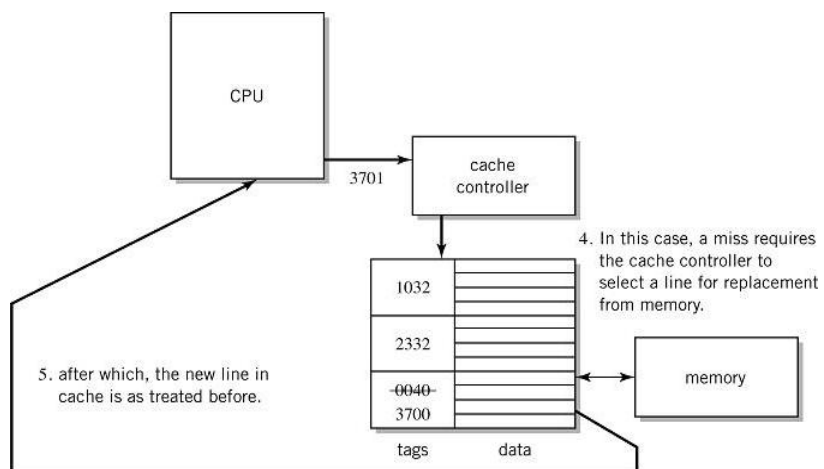


Copyright 2010-2011 John Wiley & Sons, Inc. & Robert F. Kelly

8-19



Step-by-Step Use of Cache



Copyright 2010-2011 John Wiley & Sons, Inc. & Robert F. Kelly

8-20



Performance Advantages

- Hit ratios of 90% common
- 50%+ improved execution speed
- Locality of reference is why caching works
 - Most memory references confined to small region of memory at any given time
 - Well-written program in small loop, procedure or function
 - Variables stored together
- Some processors feature pre-loading and separate instruction and caches

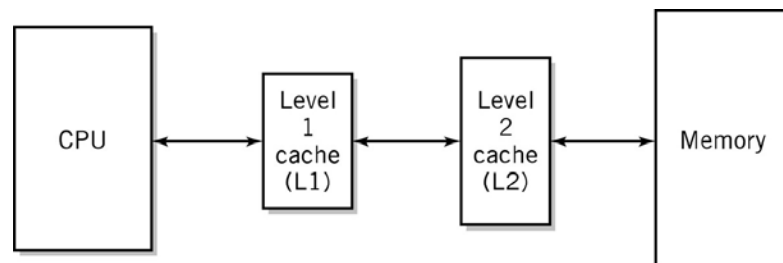
Copyright 2010-2011 John Wiley & Sons, Inc. & Robert F. Kelly

8-21



Two-level Caches

- 2-level caches can improve performance
- 2-level cache is larger with larger data lines



Copyright 2010-2011 John Wiley & Sons, Inc. & Robert F. Kelly

8-22



Multiprocessing

- Multiprocessors – multiple CPUs within a computer
- Reasons
 - Increase the processing power of a system
 - Parallel processing
- Multiprocessor system
 - Tightly coupled
 - Multicore processors - when CPUs are on a single integrated circuit

Copyright 2010-2011 John Wiley & Sons, Inc. & Robert F. Kelly

8-23



Multiprocessor Systems

- Identical access to programs, data, shared memory, I/O, etc.
- Easily extends multi-tasking, and redundant program execution
- Each processor has its own instruction counter
- Most popular configuration is Symmetrical multiprocessing (SMP)
 - Master-slave multiprocessing

Copyright 2010-2011 John Wiley & Sons, Inc. & Robert F. Kelly

8-24



Multiprocessor Advantages

- Advantages
 - Can achieve equivalent processing power at lower clock speeds, thereby reducing power consumption
 - Relatively inexpensive way to increase performance
- Disadvantages
 - Access to RAM is serialized
 - Software modifications
 - Scheduling overhead

Copyright 2010-2011 John Wiley & Sons, Inc. & Robert F. Kelly

8-25



Software Perspective

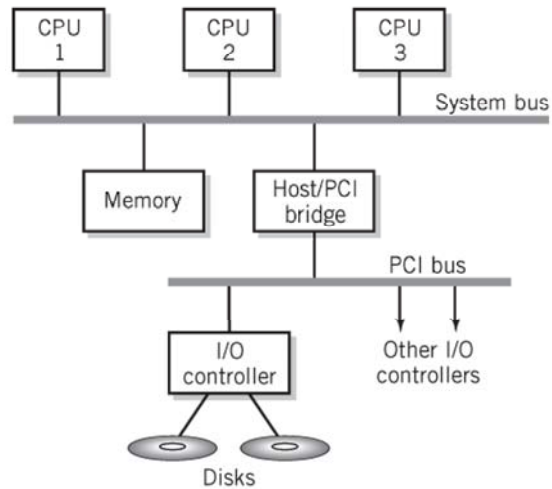
- Separate processes
 - Controlled and scheduled by OS
 - Good performance for modest numbers of processes
 - Little advantage for single applications
- Multi-threading
 - Smoother operation for user
 - Effective handling of interrupts
 - Requires application software mods (may be more prevalent as multi-cores increase)

Copyright 2010-2011 John Wiley & Sons, Inc. & Robert F. Kelly

8-26



Typical Multiprocessing System Configuration



Copyright 2010-2011 John Wiley & Sons, Inc. & Robert F. Kelly

8-27



Multicore Processors

- iPhone A4 processor
- ARM Cortex
- AMD Athlon, Phenom, Opteron, etc.
- Intel Itanium, Xeon, Core i7, Pentium III, etc.
- Sun UltraSPARC
- IBM System z

Copyright 2010-2011 John Wiley & Sons, Inc. & Robert F. Kelly

8-28



RISC Vs. CISC

- Reduced Instruction Set Computing
- Complex Instruction Set Computing
- RISC advocates a limited set of simple instructions, all of the same length
- RISC ISA usually has few memory instructions
- RISC processors require fewer transistors
- Distinction between the 2 types is now less pronounced
- Examples
 - Intel x86 – CISC
 - ARM - RISC

Copyright 2010-2011 John Wiley & Sons, Inc. & Robert F. Kelly

7-29



SIMD

- Single Instruction stream, Multiple Data stream
- Executes the same instruction simultaneously on multiple data streams
- Includes array processors and vector processors
- Useful for certain operations (e.g., 3D transformations)

Copyright 2010-2011 John Wiley & Sons, Inc. & Robert F. Kelly

10-30



SoC

- System on a Chip
- Integrating all components of a computer into a single integrated circuit
- Includes
 - Microprocessor
 - Memory
 - Clock
 - Power
 - Peripherals

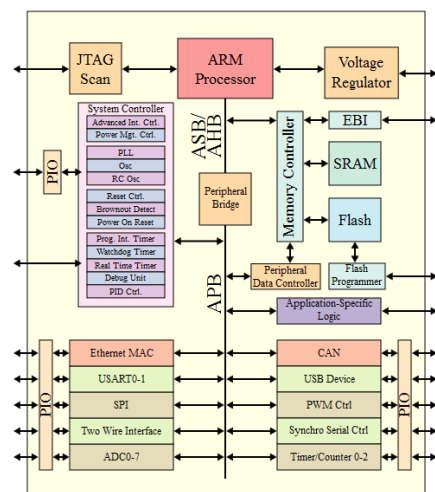
Copyright 2010-2011 John Wiley & Sons, Inc. & Robert F. Kelly

8-31



SoC Example

- ARM



Copyright 2010-2011 John Wiley & Sons, Inc. & Robert F. Kelly

8-32



MicroController

- Typically under 100K of RAM
- Often single-chip-systems;
- Usually a dedicated application