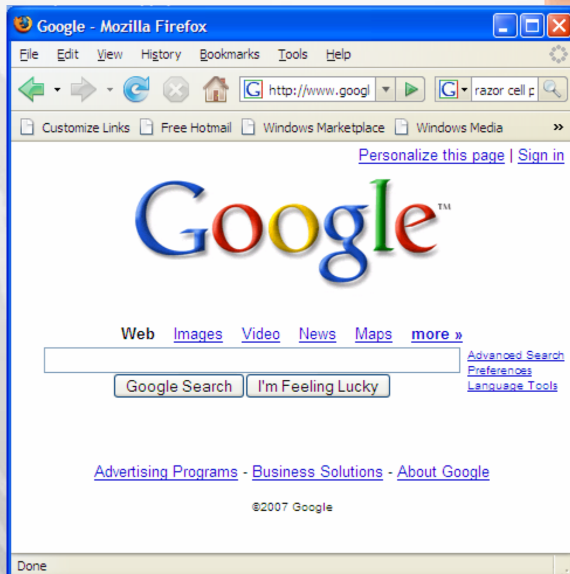
Computer Architecture

Introduction

Background

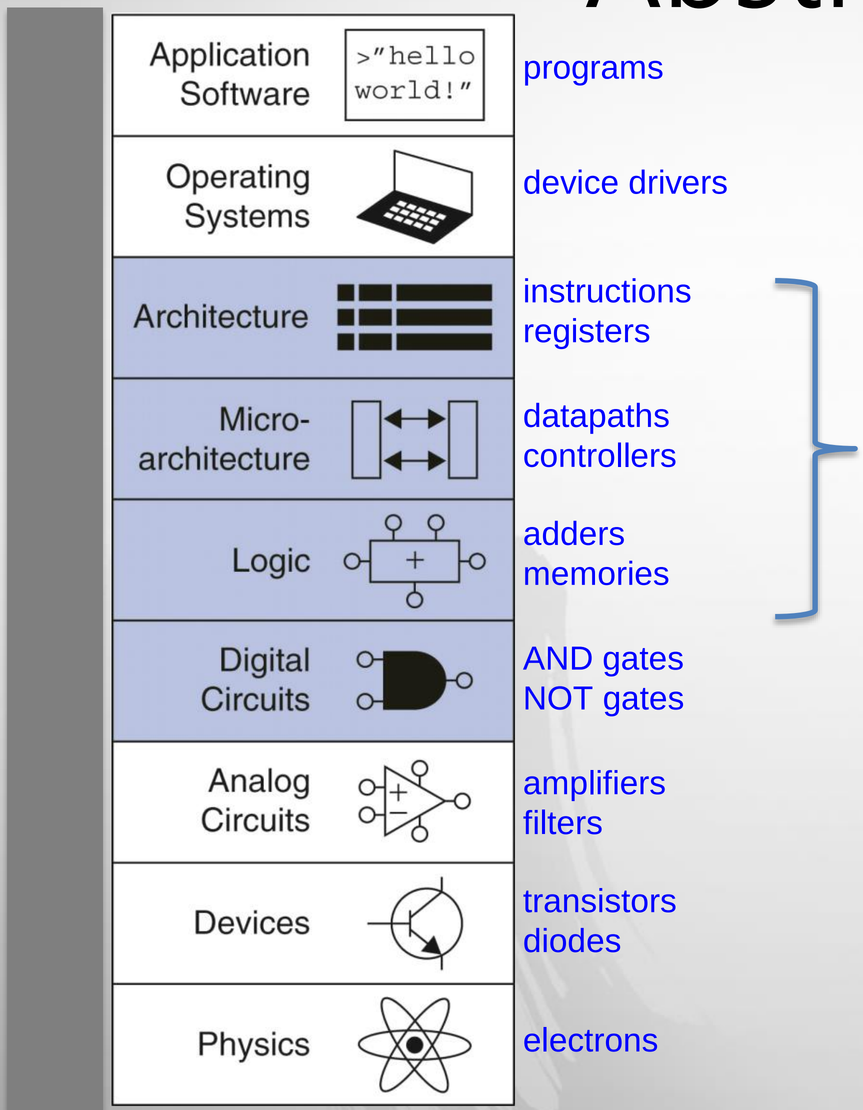
- Microprocessors have revolutionized our world
  - Cell phones, Internet, rapid advances in medicine, etc.
- The semiconductor industry has grown from $21 billion in 1985 to $300 billion in 2011



Goal

- Purpose of course:
  - Understand what's under the hood of a computer
  - Learn the principles of digital design
  - Learn to systematically debug increasingly complex designs
  - Design and build a microprocessor

Abstraction



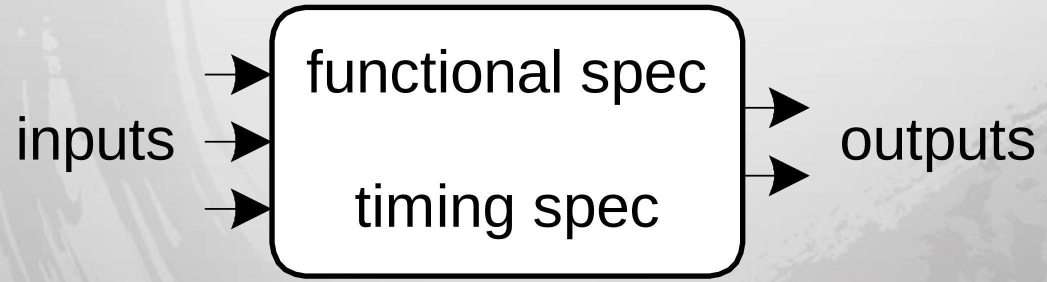
- Hiding details when they aren't important

Prerequisites (Ch 1-4)

- Logic circuits
- Boolean algebra
- Synchronous circuit
- Finite state machines

Logic Circuits

- Combinational Logic
  - Memoryless
  - Outputs determined by current values of inputs
- Sequential Logic
  - Has memory
  - Outputs determined by previous and current values of inputs



Number Systems

- Decimal numbers

1000's column
100's column
10's column
1's column

$$5374_{10} = 5 \times 10^3 + 3 \times 10^2 + 7 \times 10^1 + 4 \times 10^0$$

five thousands | three hundreds | seven tens | four ones

- Binary numbers

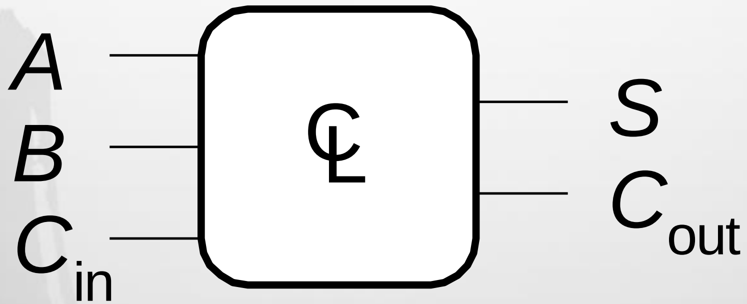
8's column
4's column
2's column
1's column

$$1101_2 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 13_{10}$$

one eight | one four | no two | one one

Boolean Functions

- Functional specification of outputs in terms of inputs
- Example: S = F(A, B, Cin)
  Cout = F(A, B, Cin)



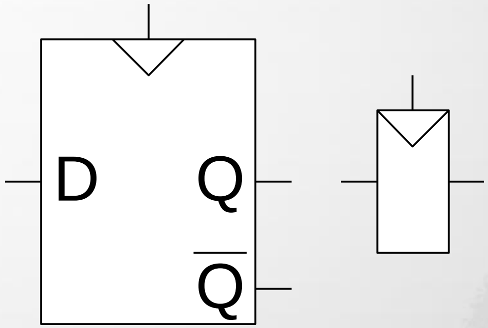
$$S = A \oplus B \oplus C_{in}$$

$$C_{out} = AB + AC_{in} + BC_{in}$$

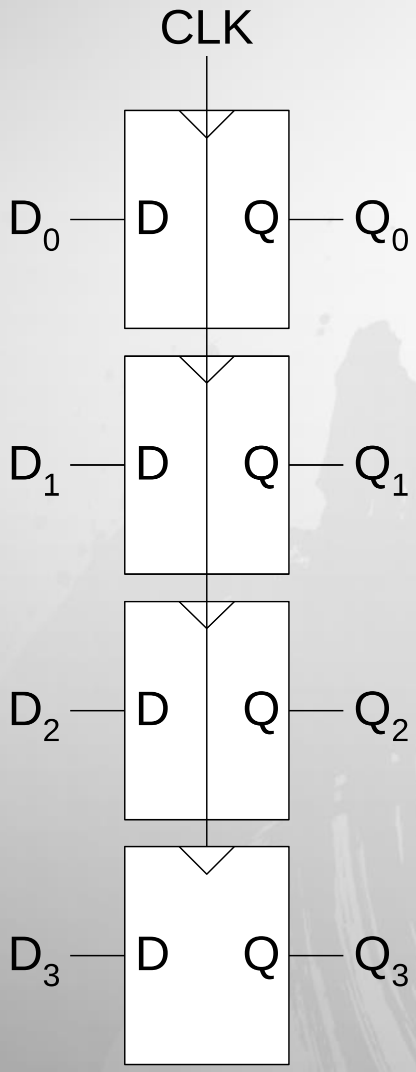
D Flip-flop

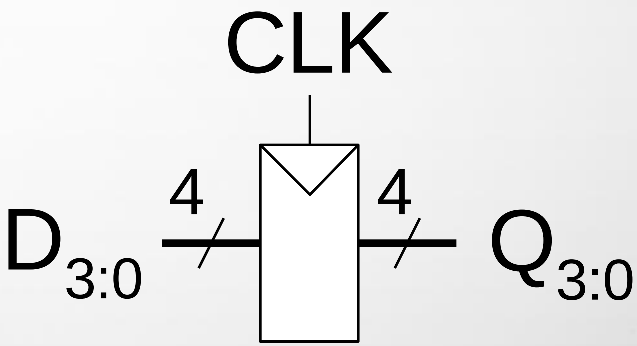
- Inputs: CLK, D
- Function
  - Samples D on rising edge of CLK
    - When CLK rises from 0 to 1, D passes through to Q
    - Otherwise, Q holds its previous value
  - Q changes only on rising edge of CLK

D Flip-Flop Symbols



Registers

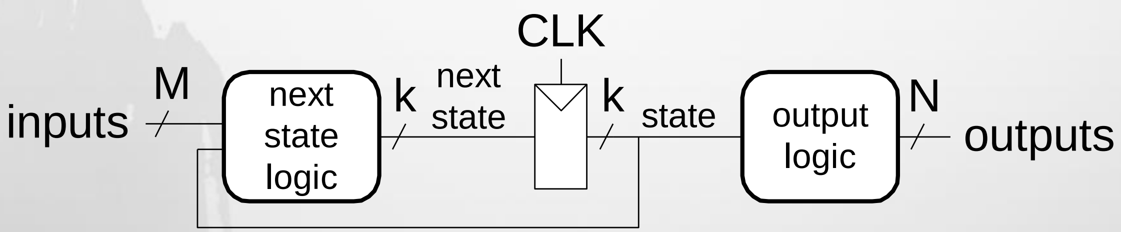




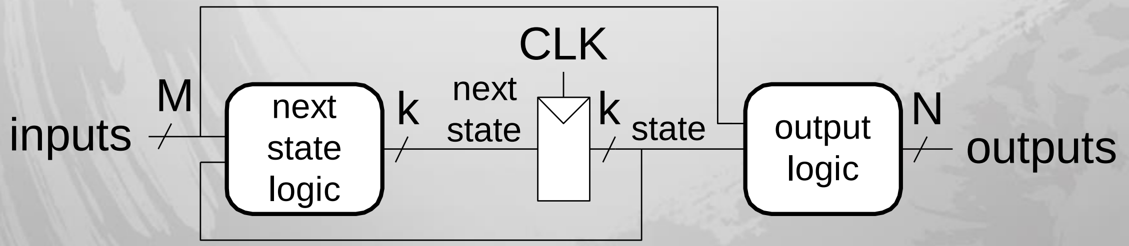
Finite State Machine

- Next state determined by current state and inputs
- Two types of finite state machines differ in output logic:
  - Moore FSM: outputs depend only on current state
  - Mealy FSM: outputs depend on current state and inputs

Moore FSM



Mealy FSM



Topics to be Covered (Ch 5-8)

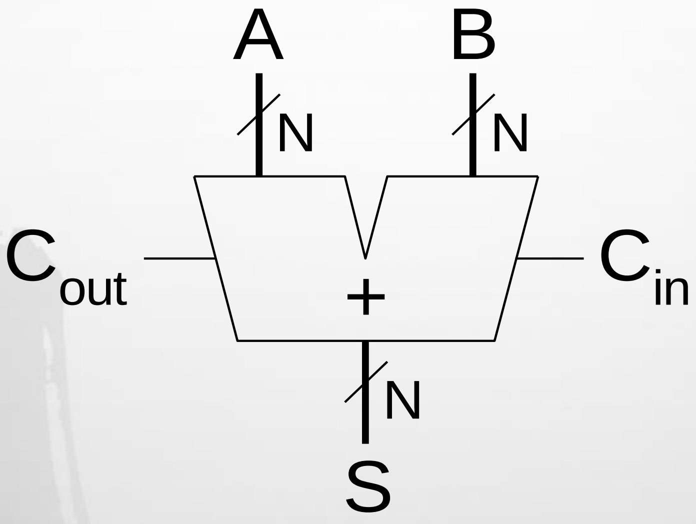
- Basic building blocks of processor
- Assembly language
- Microarchitecture of processor
- Memory system and I/O

Chapter 5 Topics

- Arithmetic Circuits
- Number Systems
- Sequential Building Blocks
- Memory Arrays
- Logic Arrays

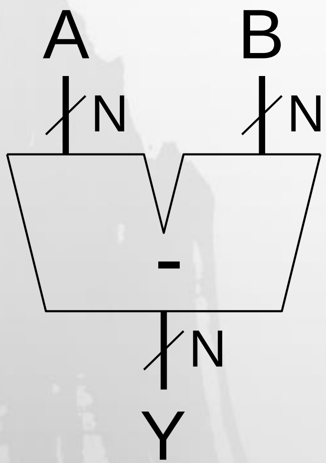
Adder

- S = A + B

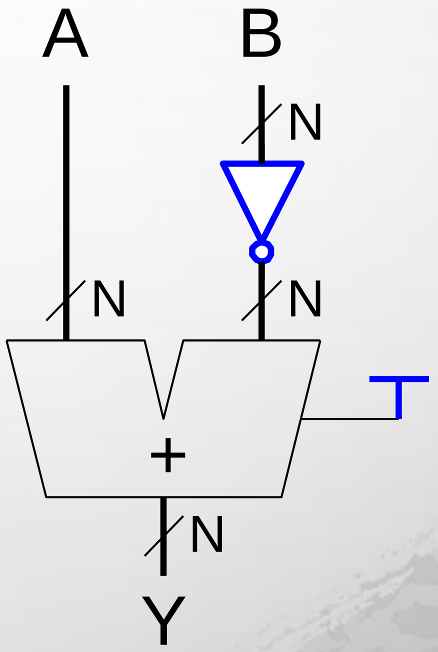


Subtractor

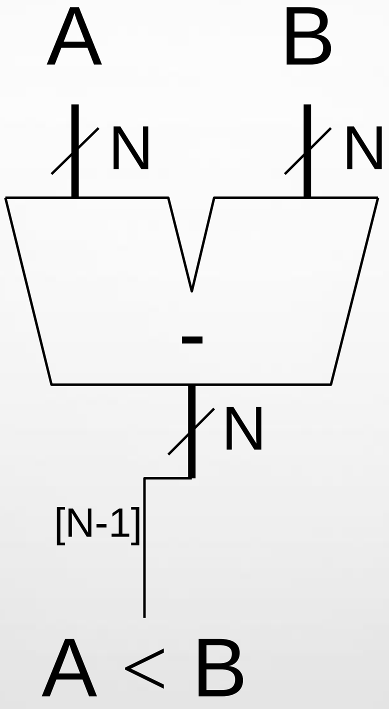
Symbol



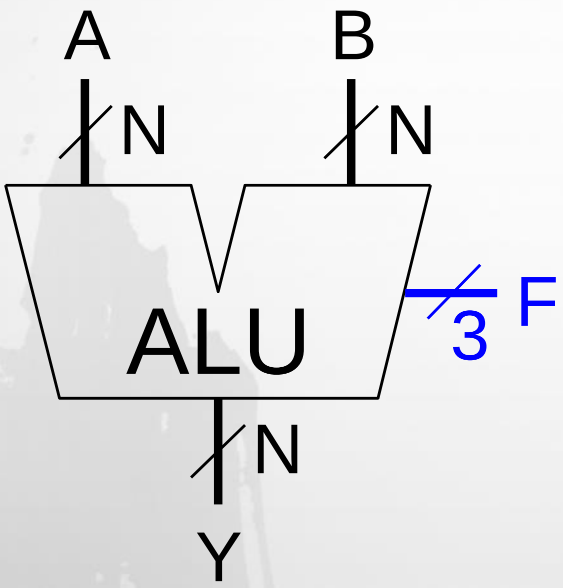
Implementation



Comparator



Arithmetic Logic Unit (ALU)



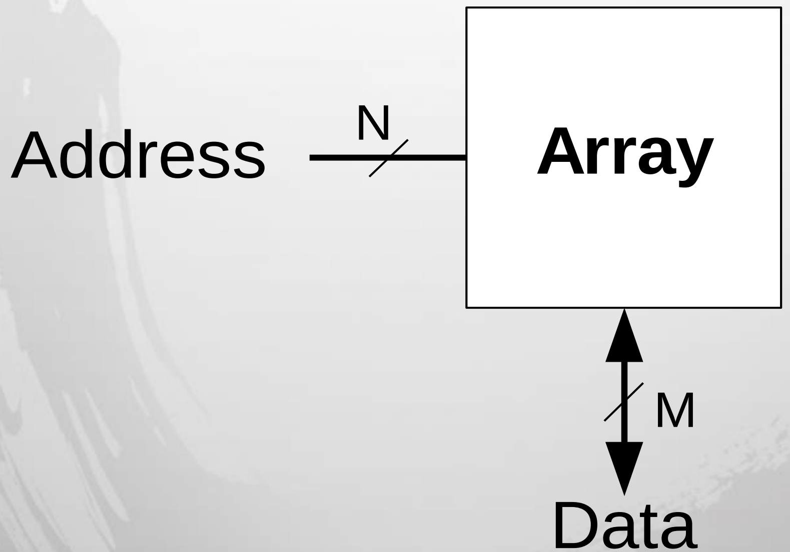
$F_{2:0}$	Function
000	A & B
001	A | B
010	A + B
011	not used
100	A & ~B
101	A | ~B
110	A - B
111	SLT

More Arithmetic Units

- Shifter
- Multiplier
- Divider
- Floating point

Memory

- Efficiently store large amounts of data
- 3 common types:
  - Dynamic random access memory (DRAM)
  - Static random access memory (SRAM)
  - Read only memory (ROM)
- M-bit data value read/ written at each unique N-bit address



Chapter 6 Topics

- Assembly Language
- Machine Language
- Programming
- Addressing Modes
- Lights, Camera, Action: Compiling, Assembling, & Loading
- Odds and Ends

Assembly Language

- Instructions: commands in a computer's language
  - **Assembly language**: human-readable format of instructions
  - **Machine language**: computer-readable format (1's and 0's)
- MIPS architecture:
  - Developed by John Hennessy and his colleagues at Stanford and in the 1980's.
  - Used in many commercial systems, including Silicon Graphics, Nintendo, and Cisco

Once you've learned one architecture, it's easy to learn others

Example

C Code

```
a = b + c;
```

MIPS assembly code

```
add a, b, c
```

- **add**: mnemonic indicates operation to perform
- **b, c**: source operands
(on which the operation is performed)
- **a**: destination operand
(to which the result is written)

Endian

- How to number bytes within a word?
- **Little-endian**: byte numbers start at the little (least significant) end
- **Big-endian**: byte numbers start at the big (most significant) end
- Word address is the same for big- or little-endian

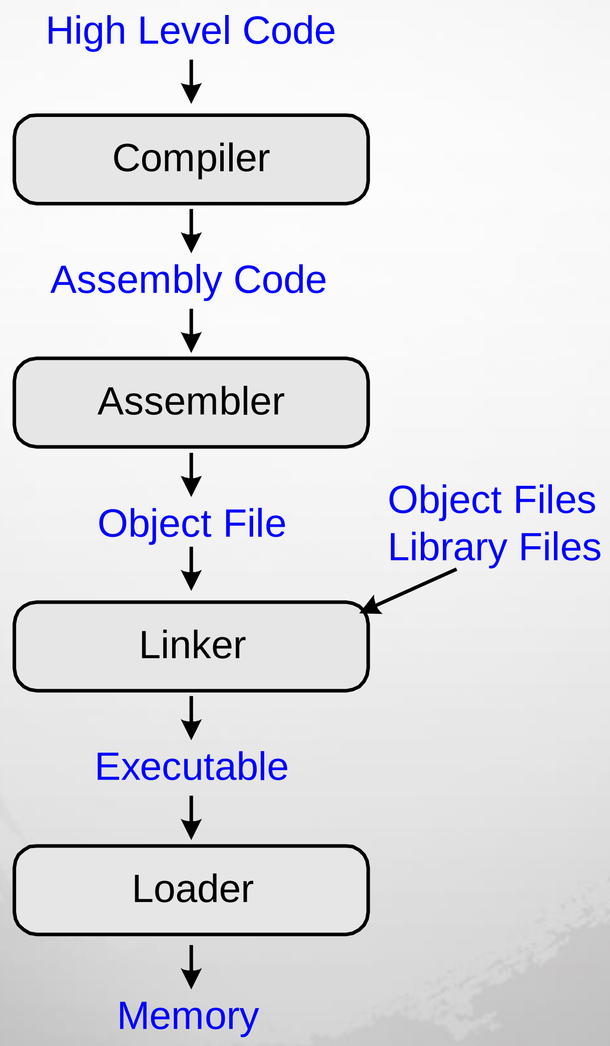
Big-Endian | **Little-Endian**

Byte Address				Word Address	Byte Address			
⋮				⋮	⋮			
C	D	E	F	C	F	E	D	C
8	9	A	B	8	B	A	9	8
4	5	6	7	4	7	6	5	4
0	1	2	3	0	3	2	1	0
MSB			LSB		MSB			LSB

Addressing Modes

- How do we address the operands?
  - Register Only
  - Immediate
  - Base Addressing
  - PC-Relative
  - Pseudo Direct

Compile & Run


High Level Code
Compiler
Assembly Code
Assembler
Object File
Object Files
Library Files
Linker
Executable
Loader
Memory

Odds & Ends

- Pseudoinstructions
- Exceptions
- Signed and unsigned instructions
- Floating-point instructions

Chapter 7 Topics

- Performance Analysis
- Single-Cycle Processor
- Multi-Cycle Processor
- Pipelined Processor
- Exceptions
- Advanced Microarchitecture

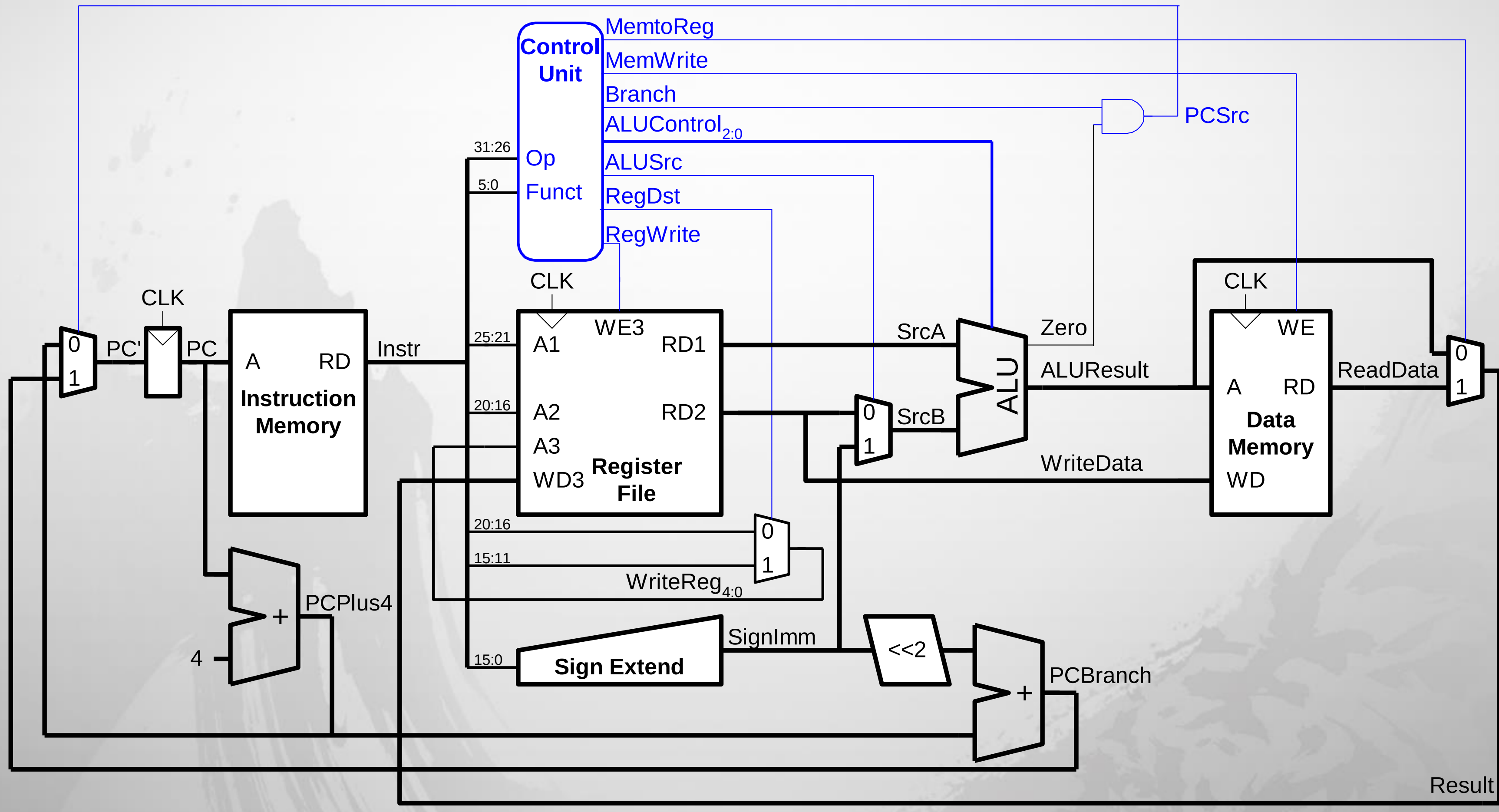
Processor Performance

- Program execution time

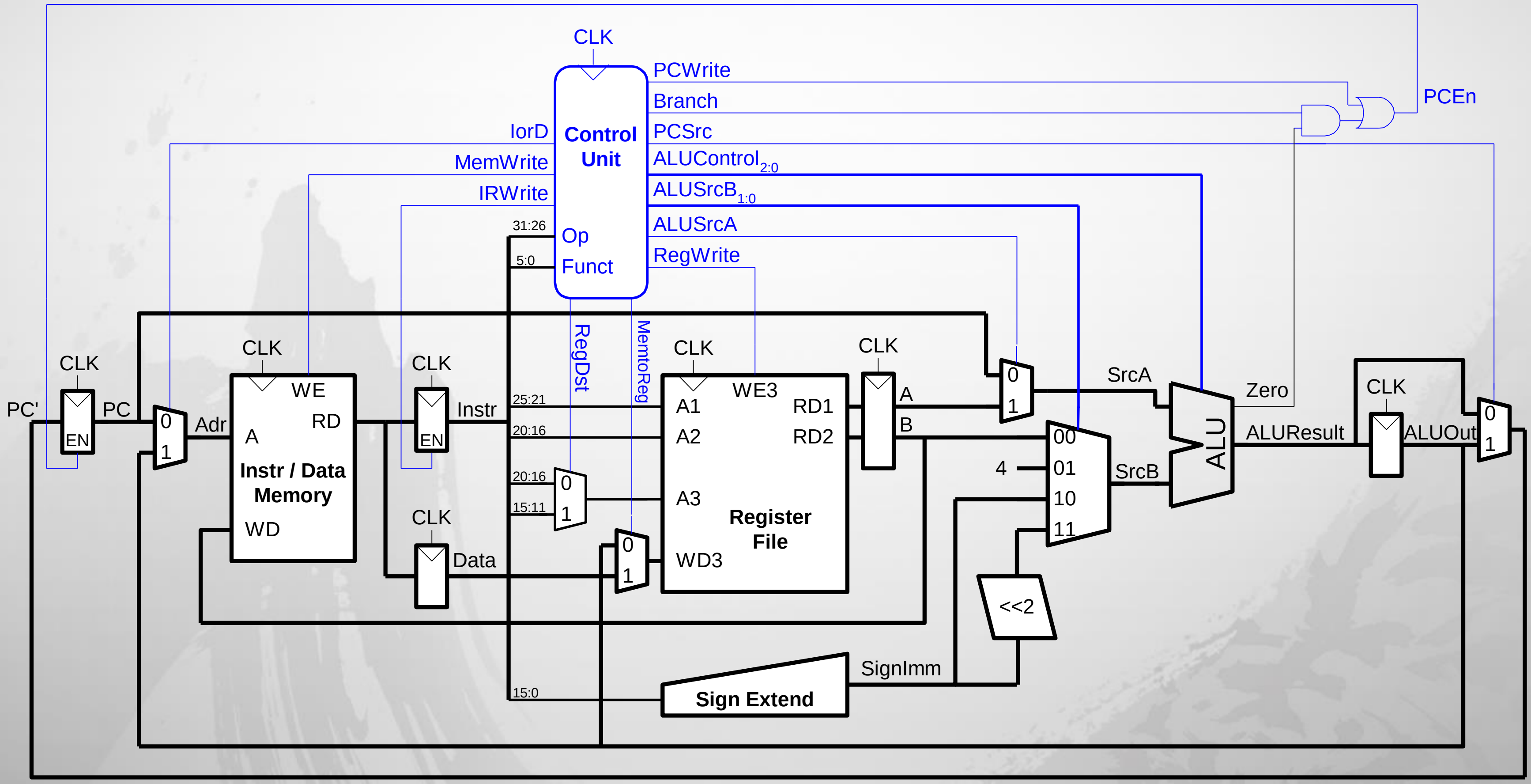
  Execution Time = (#instructions)(cycles/instruction)(seconds/cycle)

- Definitions:
  - CPI: Cycles/instruction
  - clock period: seconds/cycle
  - IPC: instructions/cycle = IPC
- Challenge is to satisfy constraints of:
  - Cost
  - Power
  - Performance

Single-Cycle Processor

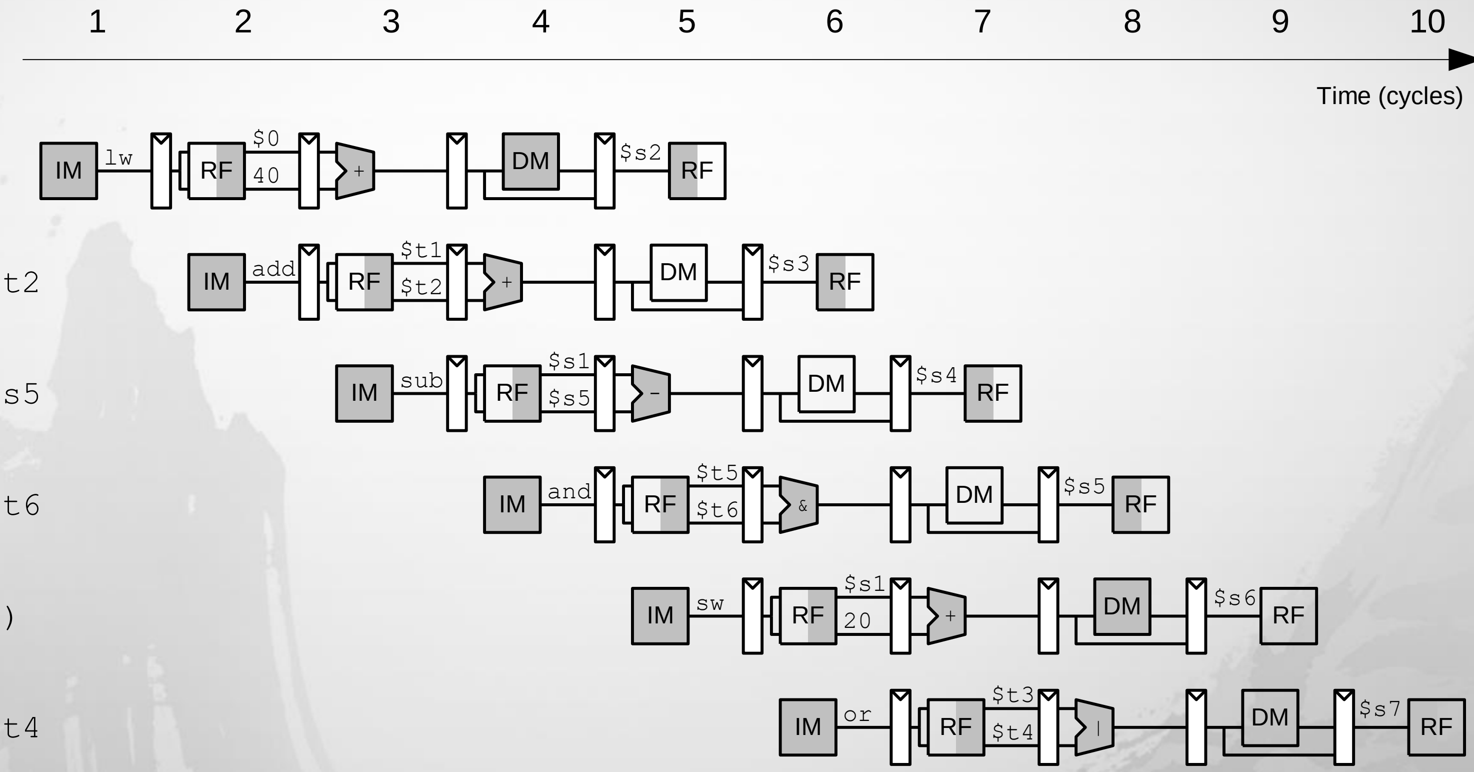


Multi-Cycle Processor

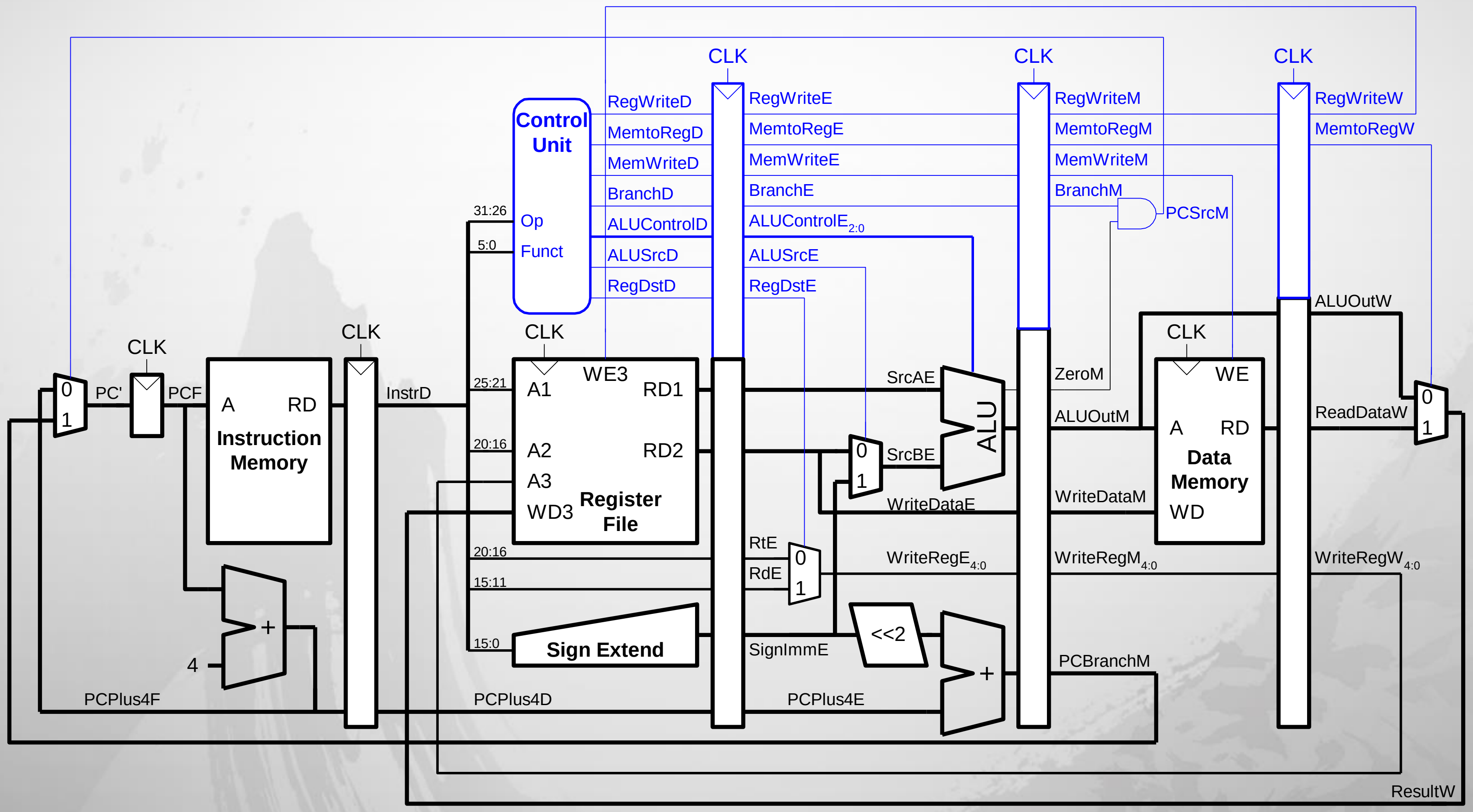


Pipelining

```
lw   $s2, 40($0)
add  $s3, $t1, $t2
sub  $s4, $s1, $s5
and  $s5, $t5, $t6
sw   $s6, 20($s1)
or   $s7, $t3, $t4
```



Pipelined Processor



Advanced Microarchitecture

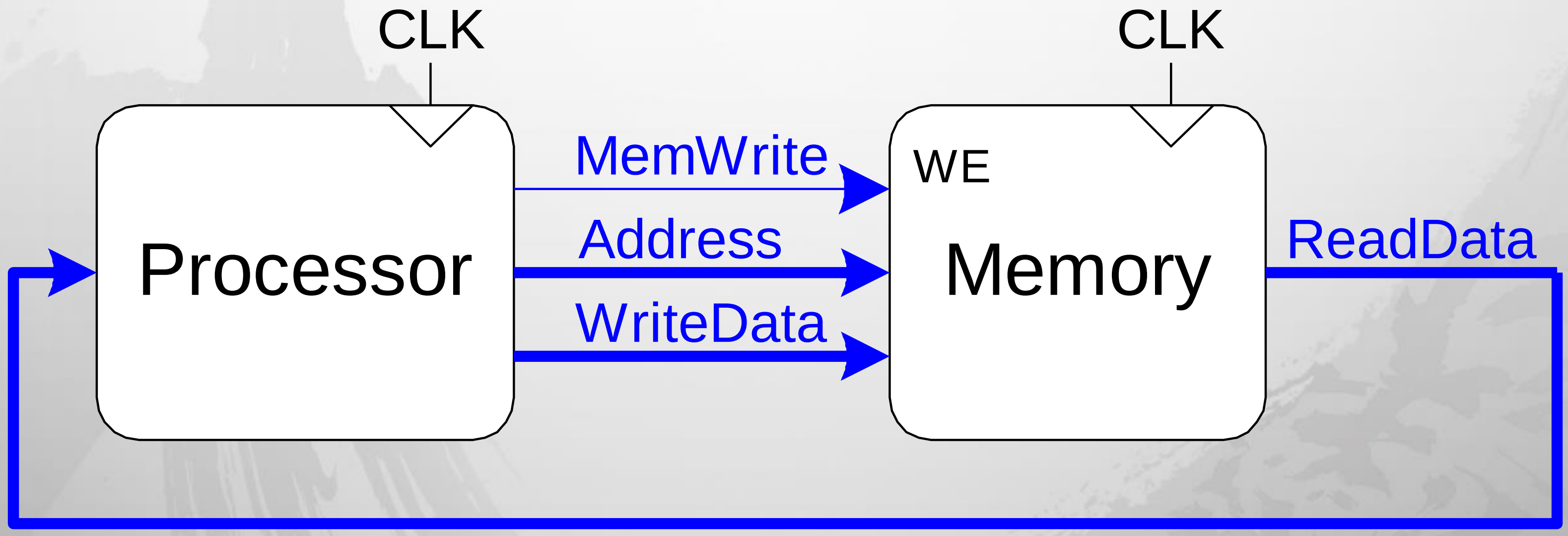
- Deep Pipelining
- Branch Prediction
- Superscalar Processors
- Out of Order Processors
- Register Renaming
- SIMD
- Multithreading
- Multiprocessors

Chapter 8 Topics

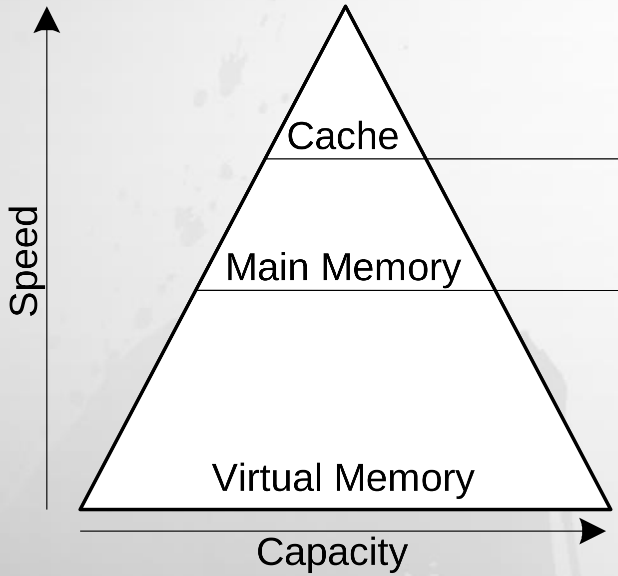
- Memory System Performance Analysis
- Caches
- Virtual Memory
- Memory-Mapped I/O

Memory System Performance

- Computer performance depends on:
  - Processor performance
  - Memory system performance



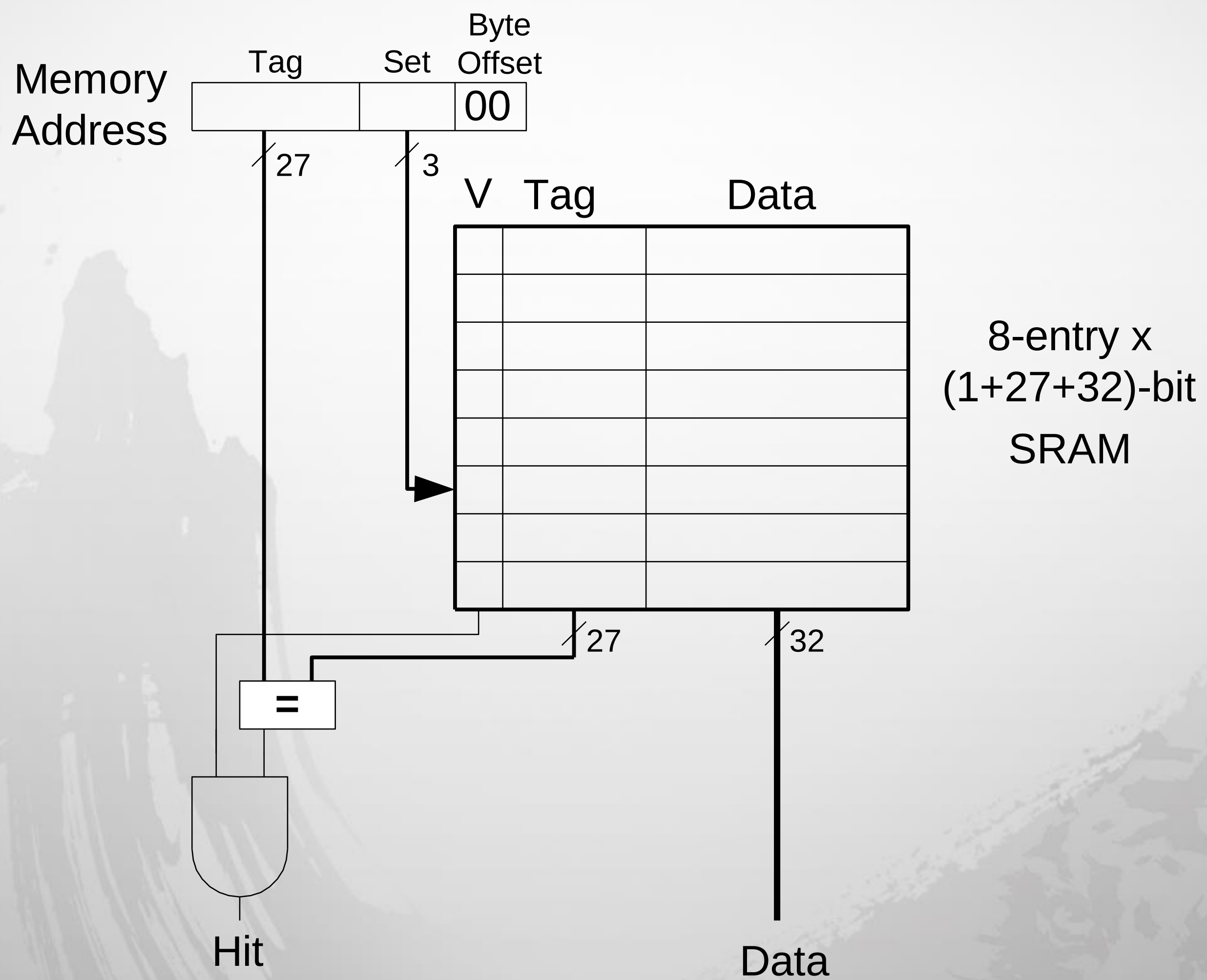
Memory Hierarchy



Technology	Price / GB	Access Time (ns)	Bandwidth (GB/s)
SRAM	$10,000	1	25+
DRAM	$10	10 - 50	10
SSD	$1	100,000	0.5
HDD	$0.1	10,000,000	0.1

- **Physical Memory:** DRAM (Main Memory)
- **Virtual Memory:** Hard drive
  - Slow, Large, Cheap

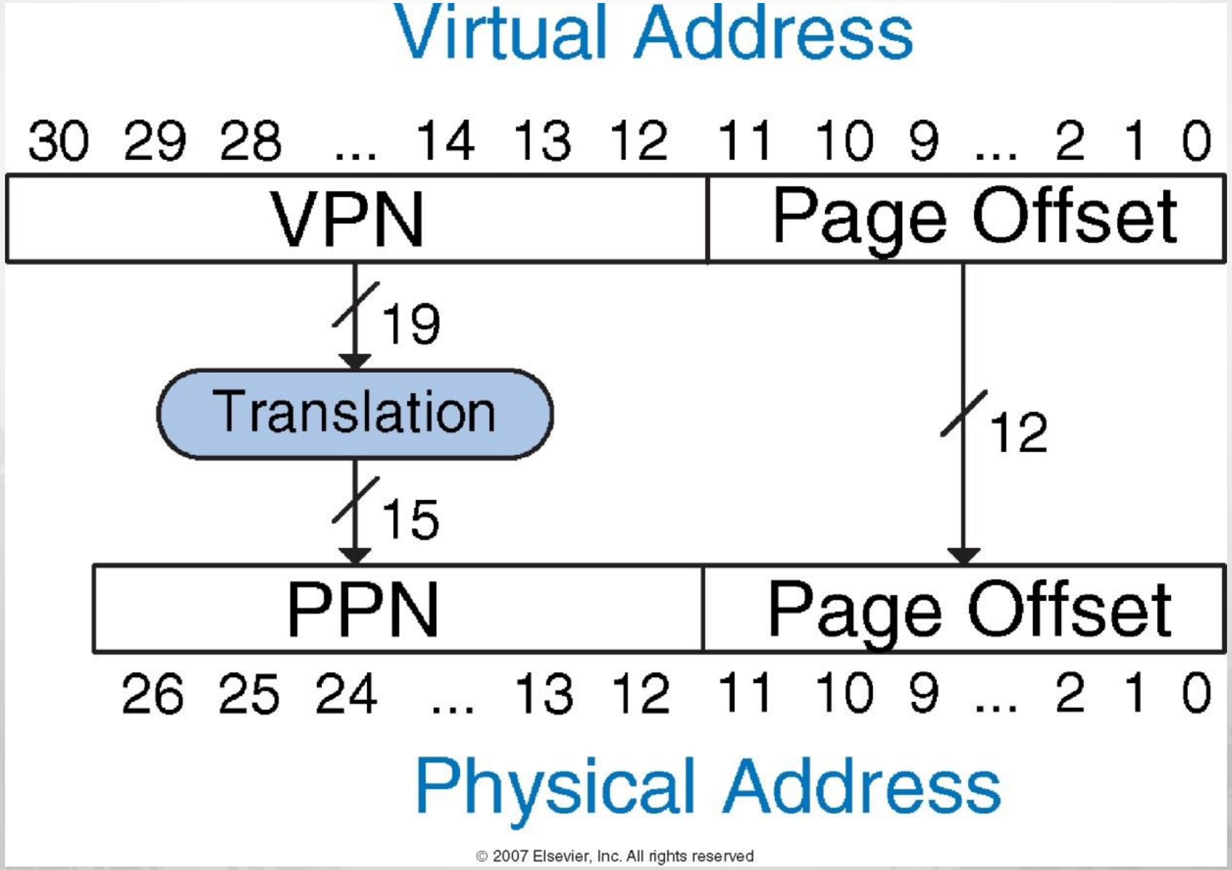
Direct Map Cache



Virtual Memory

- Virtual addresses
  - Programs use virtual addresses
  - Entire virtual address space stored on a hard drive
  - Subset of virtual address data in DRAM
  - CPU translates virtual addresses into physical addresses (DRAM addresses)
  - Data not in DRAM fetched from hard drive
- Memory Protection
  - Each program has own virtual to physical mapping
  - Two programs can use same virtual address for different data
  - Programs don’t need to be aware others are running
  - One program (or virus) can’t corrupt memory used by another

Address Translation



Memory Mapped I/O

- Processor accesses I/O devices just like memory (like keyboards, monitors, printers)
- Each I/O device assigned one or more address
- When that address is detected, data read/written to I/O device instead of memory
- A portion of the address space dedicated to I/O devices