ICS 233
Computer Architecture & Assembly Language

Lecture 2

ICS 233
Computer Architecture & Assembly Language

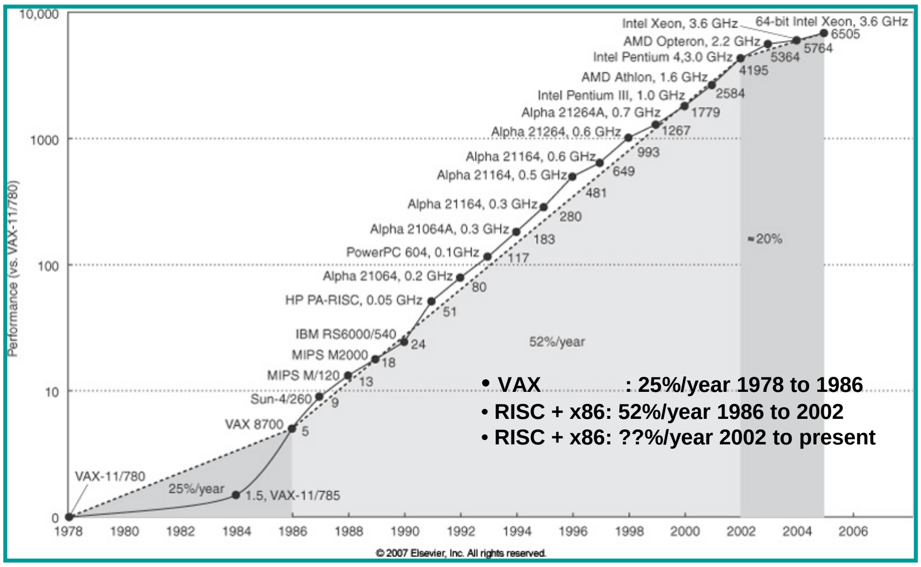
Computer Growth

Computer Growth So far…

- During 1970s, Mainframes and minicomputers dominated the industry
- Performance improved 25% to 30% per year mostly due to improved architectures and some technological developments
- Late 1970s – emergence of the microprocessor (Improvement in IC Technology) which led to higher rate of improvement – roughly 35% growth per year in performance
- Microprocessor based computers became Commercially successful due to
  - Virtual elimination of assembly language programming to reduce the need for object-code compatibility (High level Programming)
  - Creation of standardized vendor-independent operating systems, such as UNIX and its clone, Linux
  - Mass Production
- In the early 1980s, successful development of a new set of architectures called RISC (Reduced Instruction Set Computer)
- RISC-based machines focused on two critical performance techniques
  - Exploitation of instruction level parallelism (pipelining, multiple instruction issue)
  - Use of caches

Computer Growth So far…

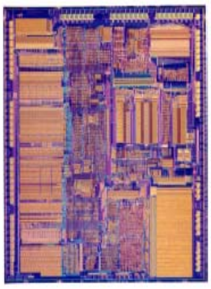
Combination of architectural & organizational enhancements and efficient use of technology improvements led to 20 years of sustained growth in performance at an annual rate of over 50%

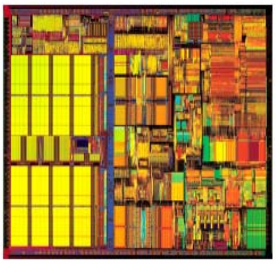


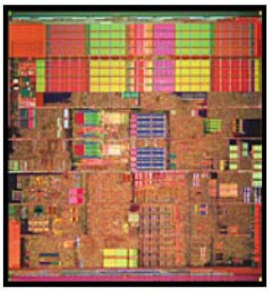
- VAX : 25%/year 1978 to 1986
- RISC + x86: 52%/year 1986 to 2002
- RISC + x86: ??%/year 2002 to present

© 2007 Elsevier, Inc. All rights reserved.

From Intel 386 to Pentium 4


Intel 386, introduced 1985
275,000 transistors, 1 micron
16 MHz clock speed


Intel Pentium III, introduced 1999
9.5M transistors, 0.25 micron
600 MHz clock speed


Intel 4 Prescott, introduced late 04
125M transistors, 0.09 micron
2.8-3.8 GHz clock speed

Processor Performance - Capacities

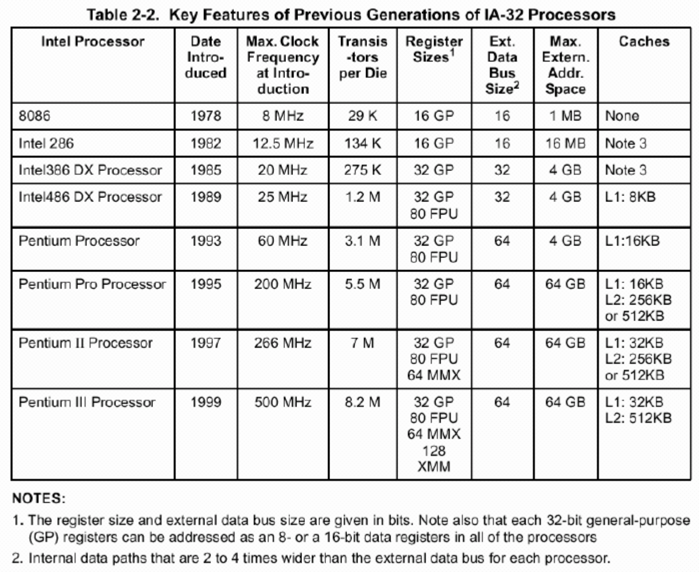


Table 2-2. Key Features of Previous Generations of IA-32 Processors

Intel Processor	Date Intro-duced	Max. Clock Frequency at Intro-duction	Transis-tors per Die	Register Sizes[1]	Ext. Data Bus Size[2]	Max. Extern. Addr. Space	Caches
8086	1978	8 MHz	29 K	16 GP	16	1 MB	None
Intel 286	1982	12.5 MHz	134 K	16 GP	16	16 MB	Note 3
Intel386 DX Processor	1985	20 MHz	275 K	32 GP	32	4 GB	Note 3
Intel486 DX Processor	1989	25 MHz	1.2 M	32 GP 80 FPU	32	4 GB	L1: 8KB
Pentium Processor	1993	60 MHz	3.1 M	32 GP 80 FPU	64	4 GB	L1:16KB
Pentium Pro Processor	1995	200 MHz	5.5 M	32 GP 80 FPU	64	64 GB	L1: 16KB L2: 256KB or 512KB
Pentium II Processor	1997	266 MHz	7 M	32 GP 80 FPU 64 MMX	64	64 GB	L1: 32KB L2: 256KB or 512KB
Pentium III Processor	1999	500 MHz	8.2 M	32 GP 80 FPU 64 MMX 128 XMM	64	64 GB	L1: 32KB L2: 512KB

NOTES:

1. The register size and external data bus size are given in bits. Note also that each 32-bit general-purpose (GP) registers can be addressed as an 8- or a 16-bit data registers in all of the processors
2. Internal data paths that are 2 to 4 times wider than the external data bus for each processor.

Processor Performance - Capacities

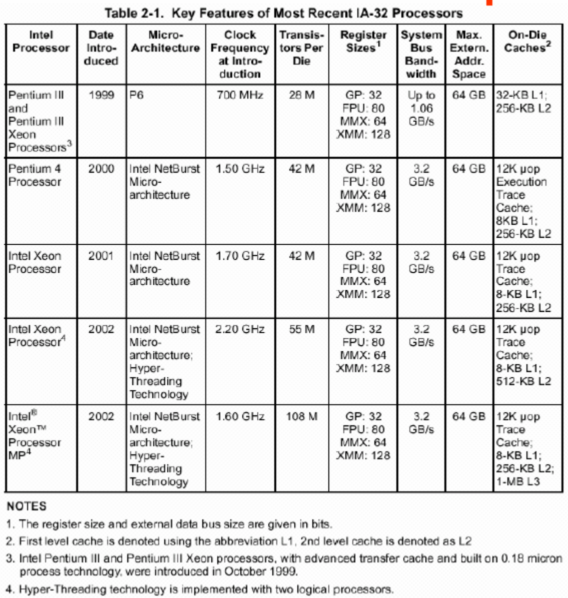


Table 2-1. Key Features of Most Recent IA-32 Processors

Intel Processor	Date Intro-duced	Micro-Architecture	Clock Frequency at Intro-duction	Transis-tors Per Die	Register Sizes[1]	System Bus Band-width	Max. Extern. Addr. Space	On-Die Caches[2]
Pentium III and Pentium III Xeon Processors[3]	1999	P6	700 MHz	28 M	GP: 32 FPU: 80 MMX: 64 XMM: 128	Up to 1.06 GB/s	64 GB	32-KB L1; 256-KB L2

Pentium 4 Processor	2000	Intel NetBurst Micro-architecture	1.50 GHz	42 M	GP: 32
					FPU: 80
					MMX: 64
					XMM: 128
						3.2 GB/s	64 GB	12K µop Execution Trace Cache;
								8KB L1;
								256-KB L2

Intel Xeon Processor	2001	Intel NetBurst Micro-architecture	1.70 GHz	42 M	GP: 32
					FPU: 80
					MMX: 64
					XMM: 128
						3.2 GB/s	64 GB	12K µop Trace Cache;
								8-KB L1;
								256-KB L2

Intel Xeon Processor[4]	2002	Intel NetBurst Micro-architecture; Hyper-Threading Technology	2.20 GHz	55 M	GP: 32
					FPU: 80
					MMX: 64
					XMM: 128
						3.2 GB/s	64 GB	12K µop Trace Cache;
								8-KB L1;
								512-KB L2

Intel® Xeon™ Processor MP[4]	2002	Intel NetBurst Micro-architecture; Hyper-Threading Technology	1.60 GHz	108 M	GP: 32
					FPU: 80
					MMX: 64
					XMM: 128
						3.2 GB/s	64 GB	12K µop Trace Cache;
								8-KB L1;
								256-KB L2;
								1-MB L3

NOTES

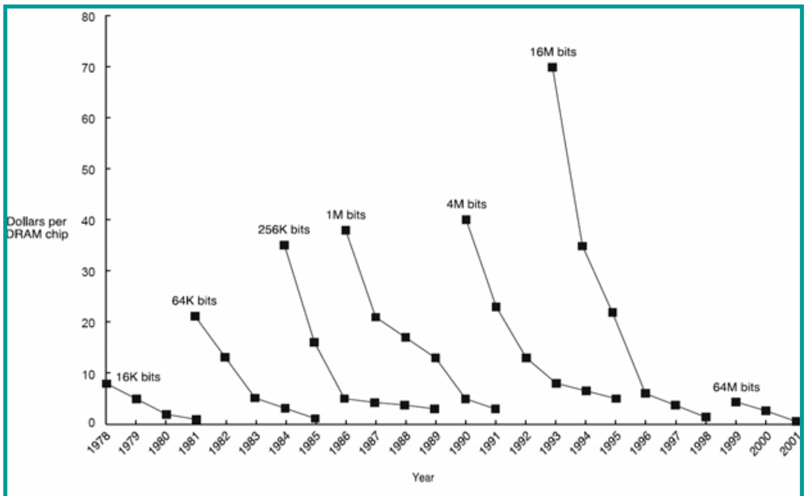
1. The register size and external data bus size are given in bits.
2. First level cache is denoted using the abbreviation L1, 2nd level cache is denoted as L2
3. Intel Pentium III and Pentium III Xeon processors, with advanced transfer cache and built on 0.18 micron process technology, were introduced in October 1999.
4. Hyper-Threading technology is implemented with two logical processors.

Currently Multi-Core Architectures

Implementation Technology Trends

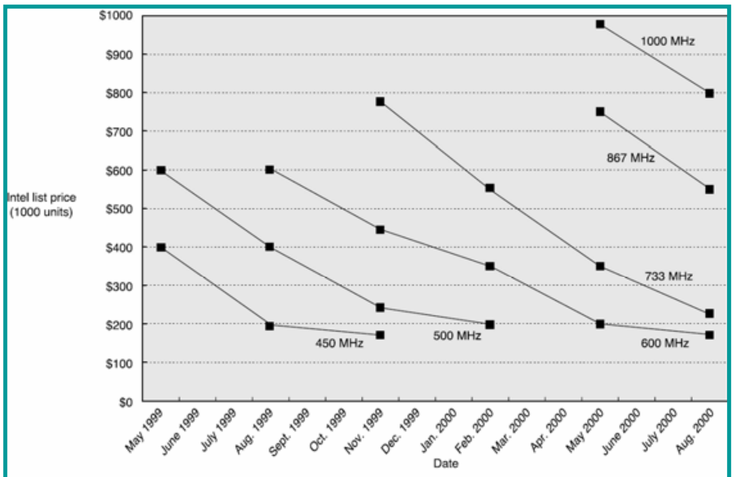
- **Four implementation technologies of interest**
  - **Integrated circuit logic**
    - **Transistor density: increases by ~35% per year**
    - **Die size: increases by ~10-20% per year**
    - **The combined effect is a growth rate in transistor count on a chip of about ~55% per year**
  - **Semiconductor DRAM**
    - **Capacity increases by ~40-60% per year**
    - **Cycle time has not decreased as much: ~33% over 10 years**
    - **Bandwidth has increased: about ~66% more over 10 years**
    - **Also, changes to the interface have helped further improve bandwidth**
  - **Magnetic disk technology**
    - **Recently, capacity improving by ~100% every year (quadrupling in two years)**
    - **Access time has improved by one-third in 10 years**
  - **Network technology**
    - **More improvements in bandwidth, less in latency**
    - **Bandwidth doubling every year in US**
    - **Gigabit Ethernet available**

DRAM Costs

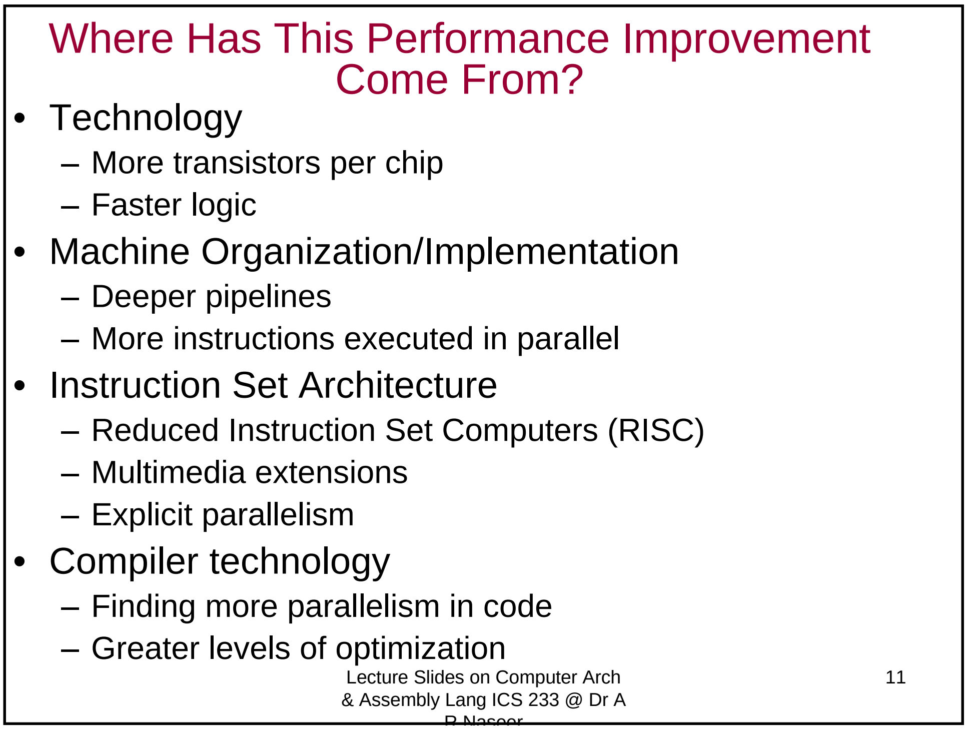


Prices of six generation of DRAMS (from 16K bits to 64Mbits)

Processor Price

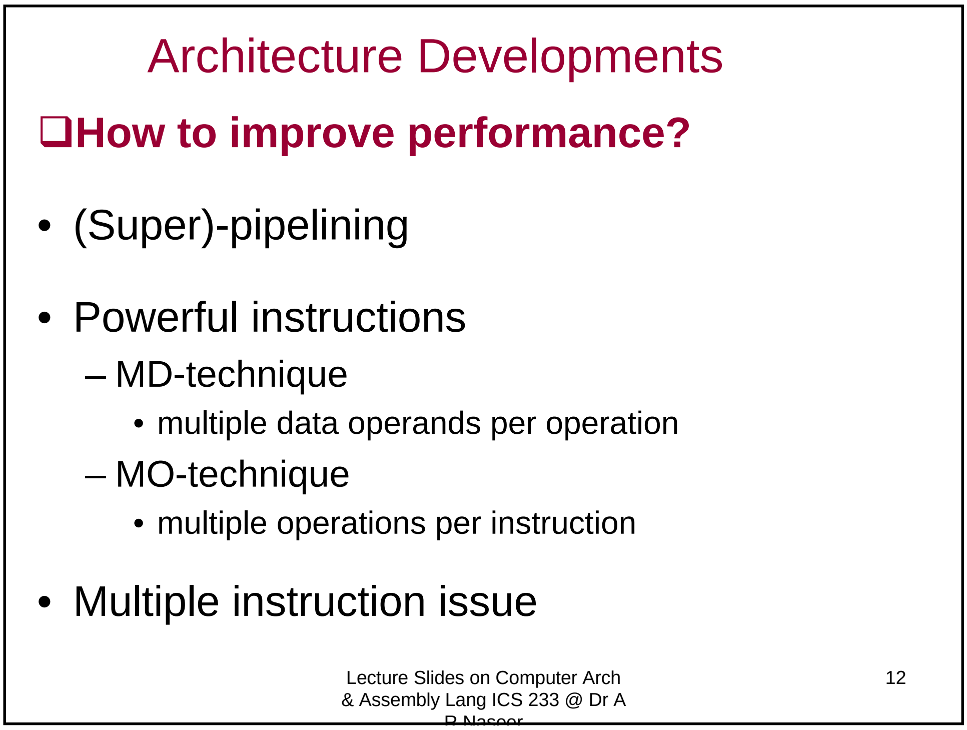


The price of an Intel Pentium III at a given frequency decreases over time as yield enhancements decrease the cost of a good die and competition forces price reductions



Where Has This Performance Improvement Come From?

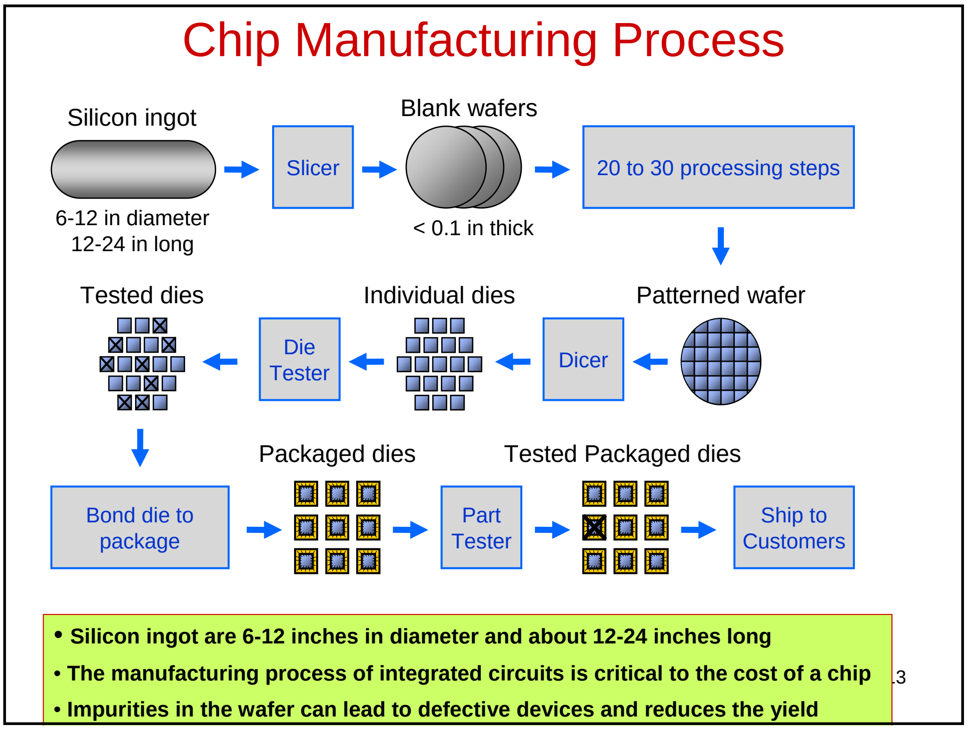
- Technology
  - More transistors per chip
  - Faster logic
- Machine Organization/Implementation
  - Deeper pipelines
  - More instructions executed in parallel
- Instruction Set Architecture
  - Reduced Instruction Set Computers (RISC)
  - Multimedia extensions
  - Explicit parallelism
- Compiler technology
  - Finding more parallelism in code
  - Greater levels of optimization



Architecture Developments

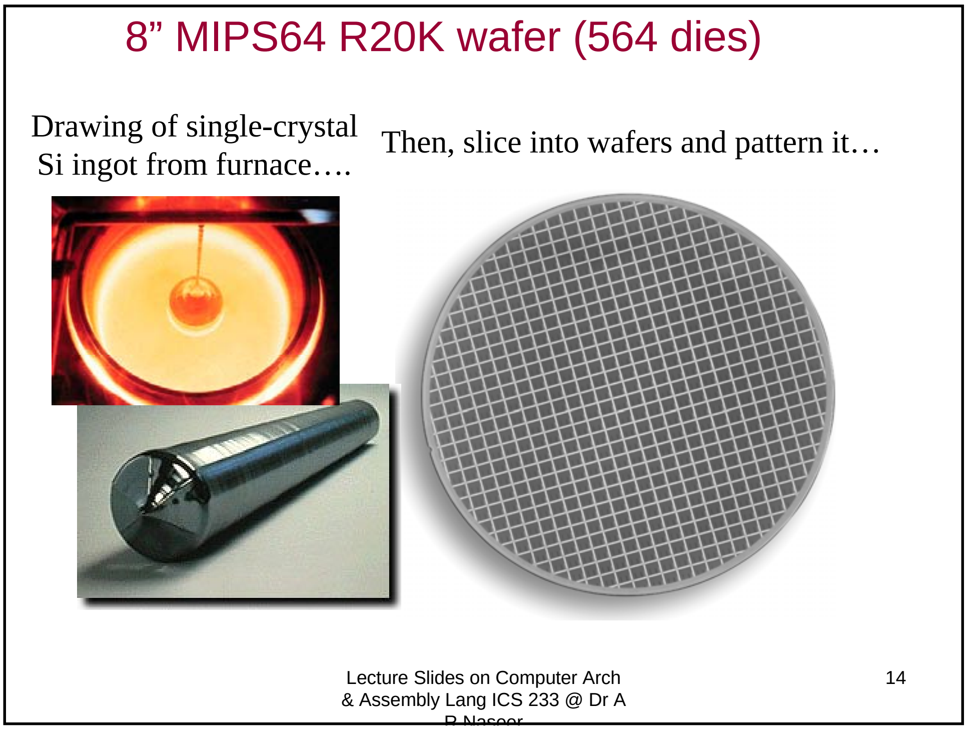
❑**How to improve performance?**

- (Super)-pipelining
- Powerful instructions
  - MD-technique
    - multiple data operands per operation
  - MO-technique
    - multiple operations per instruction
- Multiple instruction issue



Chip Manufacturing Process

- Silicon ingot are 6-12 inches in diameter and about 12-24 inches long
- The manufacturing process of integrated circuits is critical to the cost of a chip
- Impurities in the wafer can lead to defective devices and reduces the yield



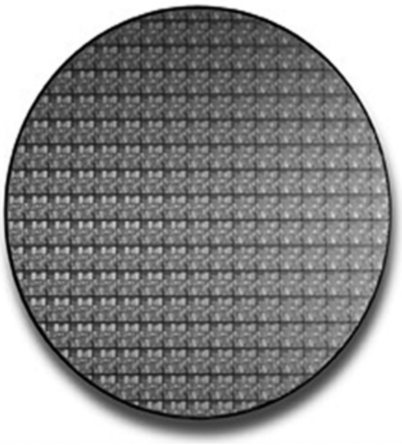
8” MIPS64 R20K wafer (564 dies)

Drawing of single-crystal Si ingot from furnace….

Then, slice into wafers and pattern it…

Wafer of Pentium 4 Processors

- 8 inches (20 cm) in diameter
- Die area is 250 mm^2
  - About 16 mm per side
- 55 million transistors per die
  - 0.18 µm technology
  - Size of smallest transistor
  - Improved technology uses
    - 0.13 µm and 0.09 µm
- Dies per wafer = 169
  - When yield = 100%
  - Number is reduced after testing
  - Rounded dies at boundary are useless



Integrated Circuits Costs

$$\text{IC cost} = \frac{\text{Die cost} + \text{Testing cost} + \text{Packaging cost}}{\text{Final test yield}}$$

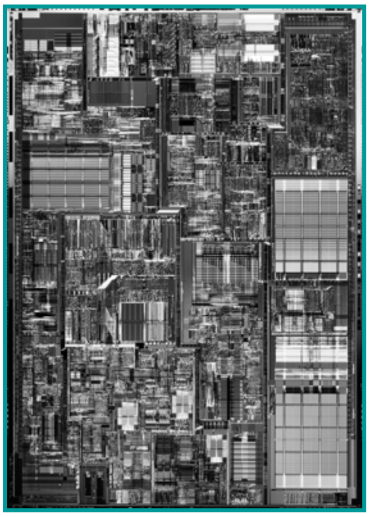
$$\text{Die cost} = \frac{\text{Wafer cost}}{\text{Dies per Wafer} * \text{Die yield}}$$

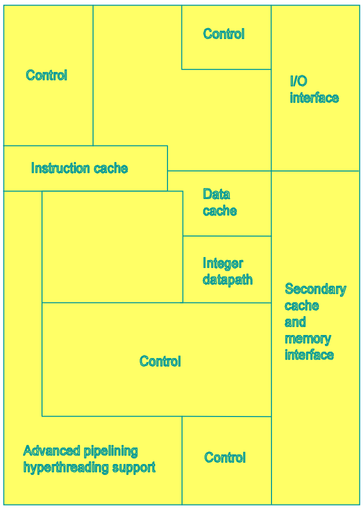
Die yield: fraction of good dies on a wafer
= Number of Good Dies / Total Number of Dies

Final test yield: fraction of packaged dies which pass the final testing state

Read Section 1.4 & refer CD for additional details

Inside the Pentium 4 Processor Chip





Control

Control

I/O interface

Instruction cache

Data cache

Integer datapath

Secondary cache and memory interface

Control

Advanced pipelining hyperthreading support

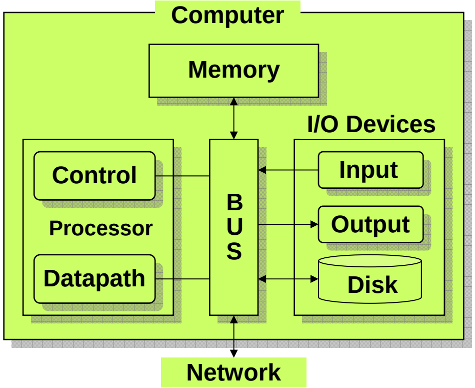
Control

ICS 233
Computer Architecture & Assembly Language

Components of a Computer System

Components of a Computer System

- **Processor**
  - Datapath
  - Control
- **Memory & Storage**
  - Main Memory
  - Disk Storage
- **Input devices**
- **Output devices**



Computer

Memory

I/O Devices

Control

Processor

Datapath

B U S

Input

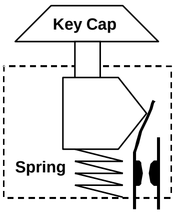
Output

Disk

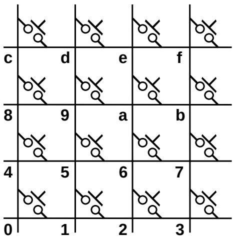
Network

- **Bus:** Interconnects processor to memory and I/O
- **Network:** newly added component for communication

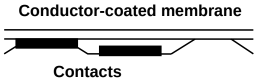
Input Devices



Mechanical switch

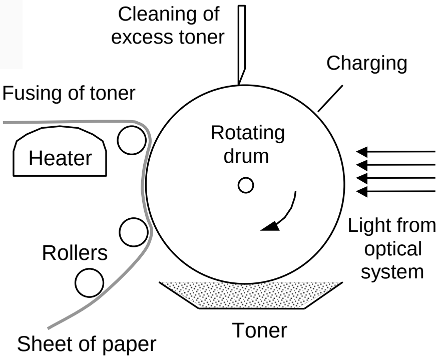


Logical arrangement of keys



Membrane switch

Output Devices

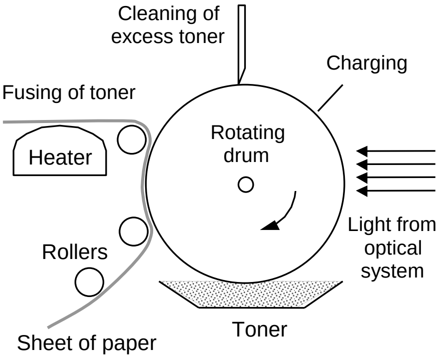


Laser printing

Output Devices

In general the principle of electrostatic laser printing is as follows:

- Charging a photoconductive selenium (or other) coated drum.
- Discharging the drum with the laser steering engine in accordance with the input image rasterized pattern. (the laser is modulated to generate a predefined pixel pattern on the face of the drum - the focal plane).
- The rotating drum attracts toner to the charged pattern (latent image) generated by the laser.
- The toner is transferred from the drum to the moving paper to generate a full image.
- The paper carrying the toner moves through the heater to fuse the toner to a fine non-erasable image.



Laser printing

Memory Devices

- **Volatile Memory Devices**
  - RAM = Random Access Memory
  - DRAM = Dynamic RAM
    - 1-Transistor cell + trench capacitor
    - Dense but slow, must be refreshed
    - Typical choice for main memory
  - SRAM: Static RAM
    - 6-Transistor cell, faster but less dense than DRAM
    - Typical choice for cache memory
- **Non-Volatile Memory Devices**
  - ROM = Read Only Memory
  - Flash Memory

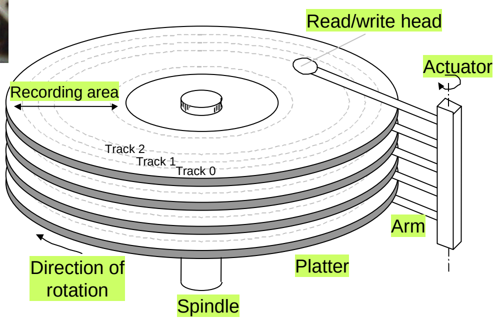
Magnetic Disk Storage

A Magnetic disk consists of a collection of platters

Provides a number of recording surfaces

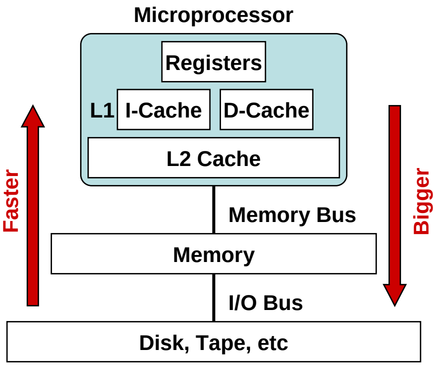
Arm provides read/write heads for all surfaces

The disk heads are connected together and move in conjunction



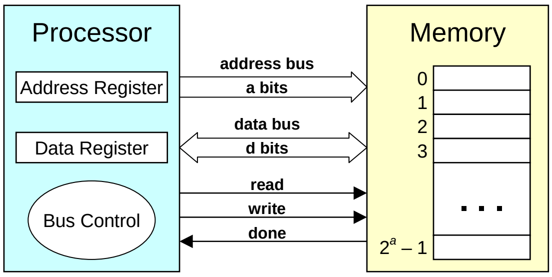
Memory Hierarchy

- Registers are at the top of the hierarchy
  - Typically few registers
  - Fastest memory element
- Level 1 Caches (8 – 64 KB)
  - I-Cache stores instructions
  - D-Cache stores data
- L2 Cache (512KB – 8MB)
  - Bigger but slower than L1
- Main Memory (1 – 2 GB)
  - Access time: 50 – 70 ns
- Disk Storage (> 200 GB)
  - Much slower than main memory, Access time = 5 ms



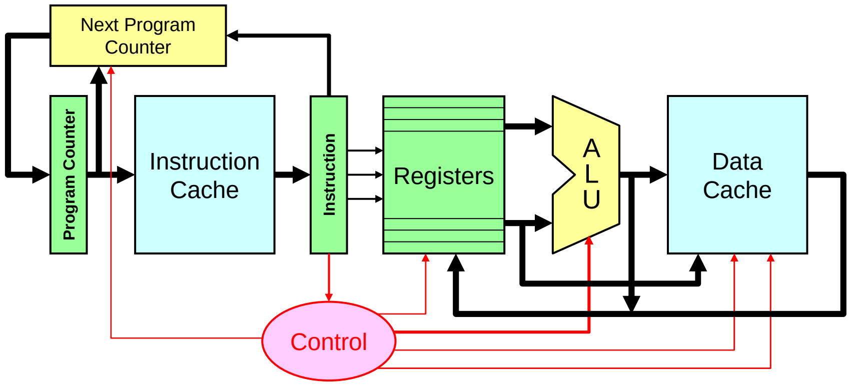
Address, Data, and Control Bus

- **Address Bus : unidirectional bus**
  - Memory address is put on address bus
  - If memory address = *a* bits then 2^a locations are addressed
- **Data Bus: bi-directional bus**
  - Data can be transferred in both directions on the data bus
- **Control Bus**
  - Signals control transfer of data
  - Read request
  - Write request
  - Done transfer



Processor

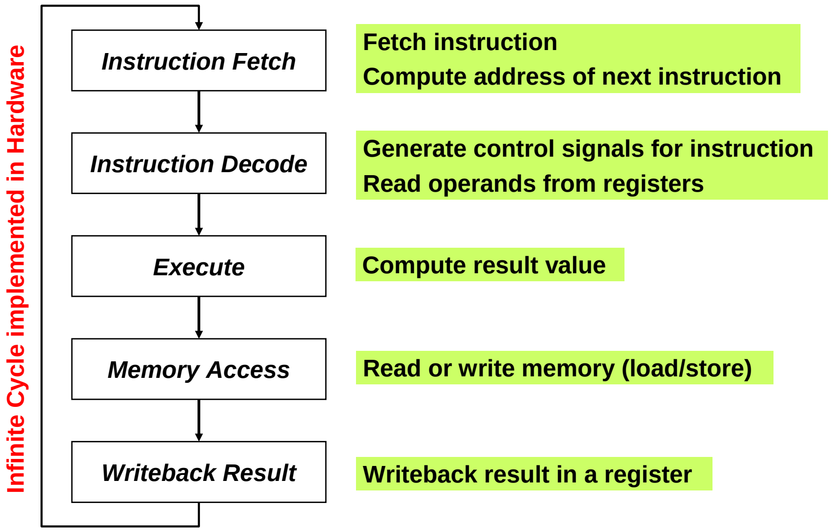
- **Datapath**: part of a processor that executes instructions
- **Control**: generates control signals for each instruction



Datapath Components

- **Program Counter (PC)**
  - Contains address of instruction to be fetched
  - Next Program Counter: computes address of next instruction
- **Instruction and Data Caches**
  - Small and fast memory containing most recent instructions/data
- **Register File**
  - General-purpose registers used for intermediate computations
- **ALU = Arithmetic and Logic Unit**
  - Executes arithmetic and logic instructions
- **Buses**
  - Used to wire and interconnect the various components

Fetch - Execute Cycle



Fetch instruction
Compute address of next instruction

Generate control signals for instruction
Read operands from registers

Compute result value

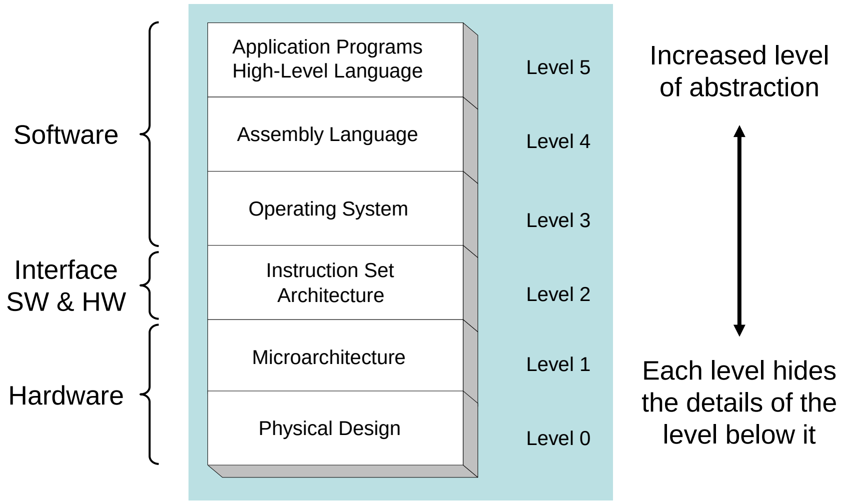
Read or write memory (load/store)

Writeback result in a register

ICS 233
Computer Architecture & Assembly Language

Programmer's View of a Computer System

Programmer’s View of a Computer System



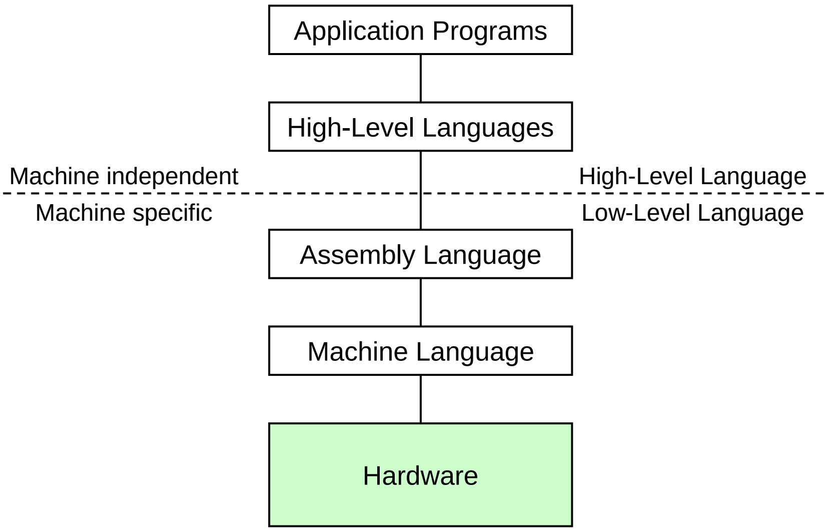
Programmer’s View of a Computer System

- **Application Programs (Level 5)**
  - Written in high-level programming languages
  - Such as Java, C++, Pascal, Visual Basic . . .
  - Programs compile into assembly language level (Level 4)
- **Assembly Language (Level 4)**
  - Instruction mnemonics are used
  - Have one-to-one correspondence to machine language
  - Calls functions written at the operating system level (Level 3)
  - Programs are translated into machine language (Level 2)
- **Operating System (Level 3)**
  - Provides services to level 4 and 5 programs
  - Translated to run at the machine instruction level (Level 2)

Programmer’s View of a Computer System

- **Instruction Set Architecture (Level 2)**
  - Interface between software and hardware
  - Specifies how a processor functions
  - Machine instructions, registers, and memory are exposed
  - Machine language is executed by Level 1 (microarchitecture)
- **Microarchitecture (Level 1)**
  - Controls the execution of machine instructions (Level 2)
  - Implemented by digital logic
- **Physical Design (Level 0)**
  - Implements the microarchitecture
  - Physical layout of circuits on a chip

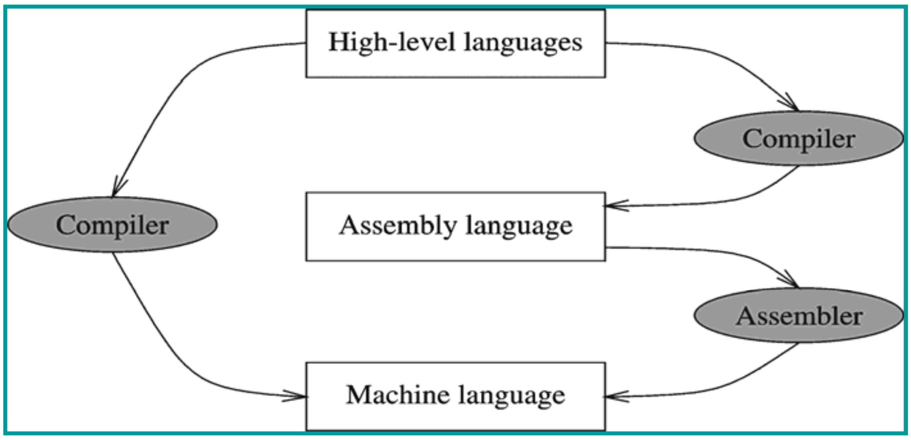
A Hierarchy of Languages

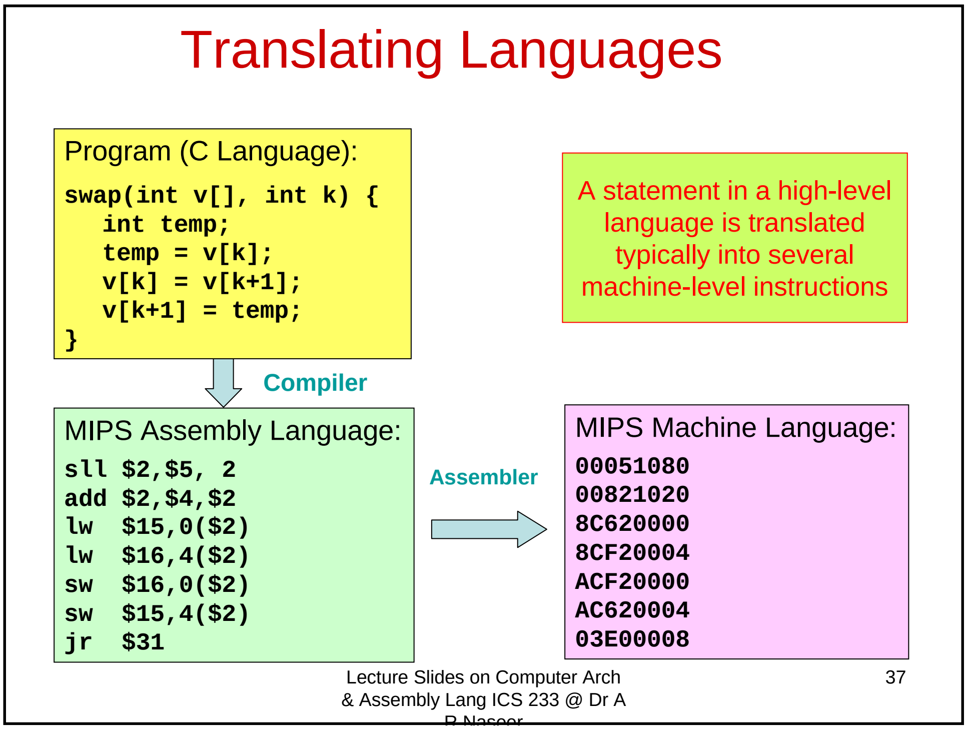


Assembly and Machine Language

- **Machine language**
  - Native to a processor: executed directly by hardware
  - Instructions consist of binary code: 1s and 0s
- **Assembly language**
  - Slightly higher-level language
  - Readability of instructions is better than machine language
  - One-to-one correspondence with machine language instructions
- **Assemblers translate assembly to machine code**
- **Compilers translate high-level programs to machine code**
  - **Either directly, or**
  - **Indirectly via an assembler**

Compiler and Assembler





Translating Languages

Program (C Language):

```
swap(int v[], int k) {
   int temp;
   temp = v[k];
   v[k] = v[k+1];
   v[k+1] = temp;
}
```

Compiler

MIPS Assembly Language:

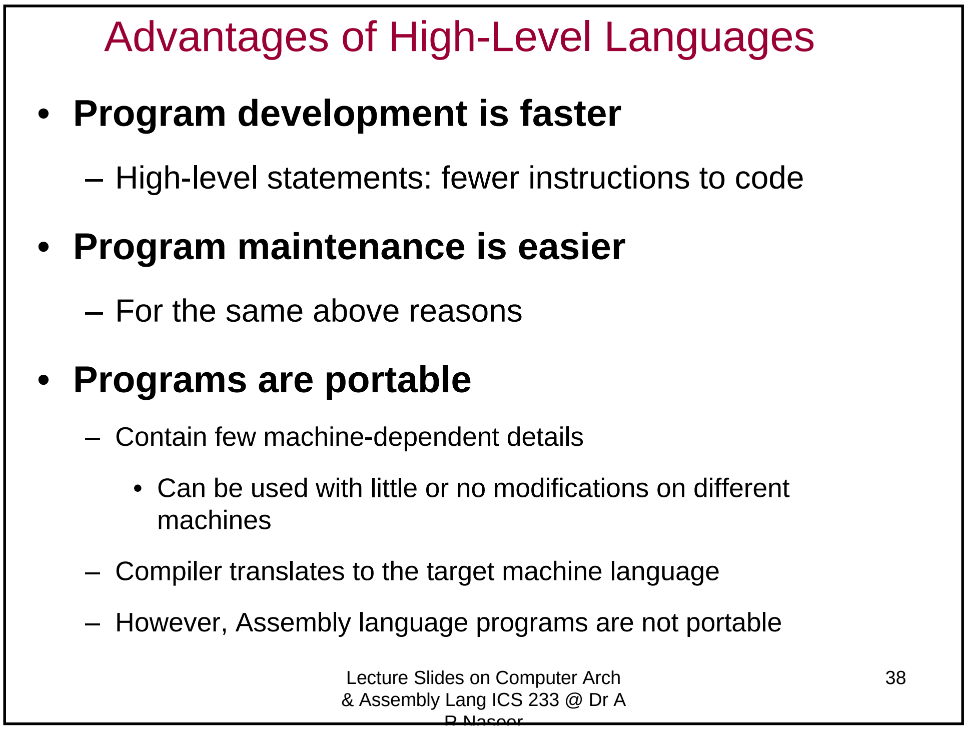
```
sll $2,$5, 2
add $2,$4,$2
lw  $15,0($2)
lw  $16,4($2)
sw  $16,0($2)
sw  $15,4($2)
jr  $31
```

Assembler

MIPS Machine Language:

```
00051080
00821020
8C620000
8CF20004
ACF20000
AC620004
03E00008
```

A statement in a high-level language is translated typically into several machine-level instructions



Advantages of High-Level Languages

- **Program development is faster**
  - High-level statements: fewer instructions to code
- **Program maintenance is easier**
  - For the same above reasons
- **Programs are portable**
  - Contain few machine-dependent details
    - Can be used with little or no modifications on different machines
  - Compiler translates to the target machine language
  - However, Assembly language programs are not portable

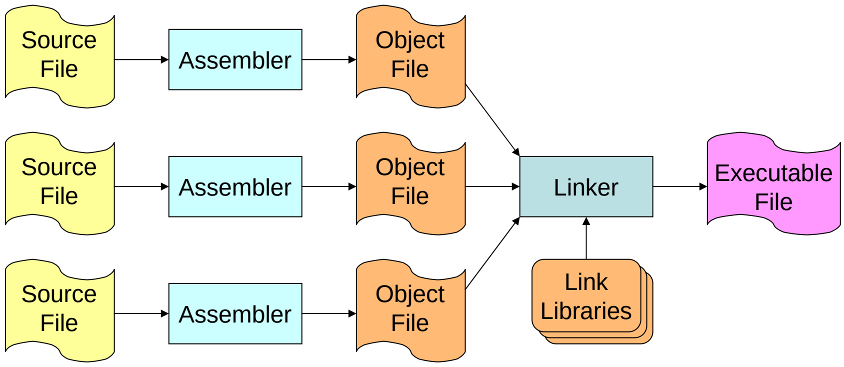
Why Learn Assembly Language?

- **Many reasons:**
  - Accessibility to system hardware
  - Space and time efficiency
  - Writing a compiler for a high-level language
- **Accessibility to system hardware**
  - Assembly Language is useful for implementing system software
  - Also useful for small embedded system applications
- **Space and Time efficiency**
  - Understanding sources of program inefficiency
  - Tuning program performance
  - Writing compact code

Assembly Language Programming Tools

- **Editor**
  - Allows you to create and edit assembly language source files
- **Assembler**
  - Converts assembly language programs into object files
  - Object files contain the machine instructions
- **Linker**
  - Combines object files created by the assembler with link libraries
  - Produces a single executable program
- **Debugger**
  - Allows you to trace the execution of a program
  - Allows you to view machine instructions, memory, and registers

Assemble and Link Process

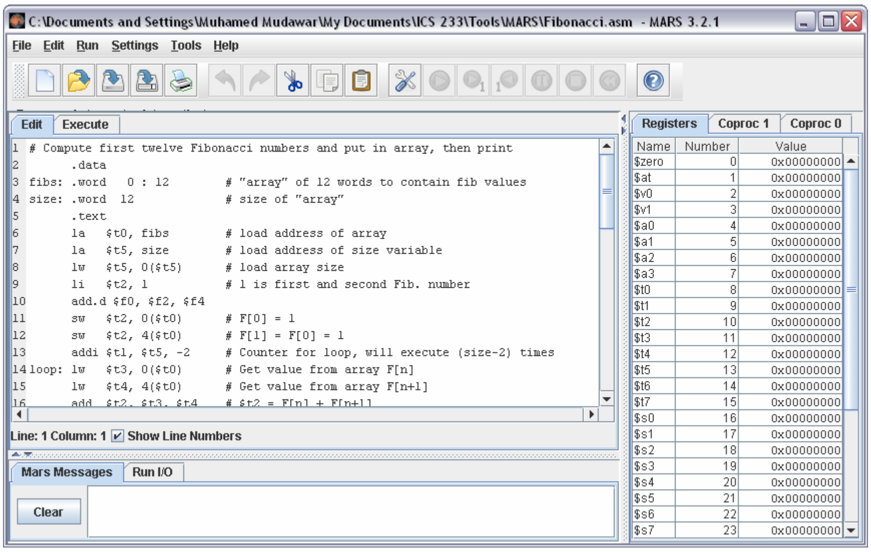


A project may consist of multiple source files

Assembler translates each source file separately into an object file

Linker links all object files together with link libraries

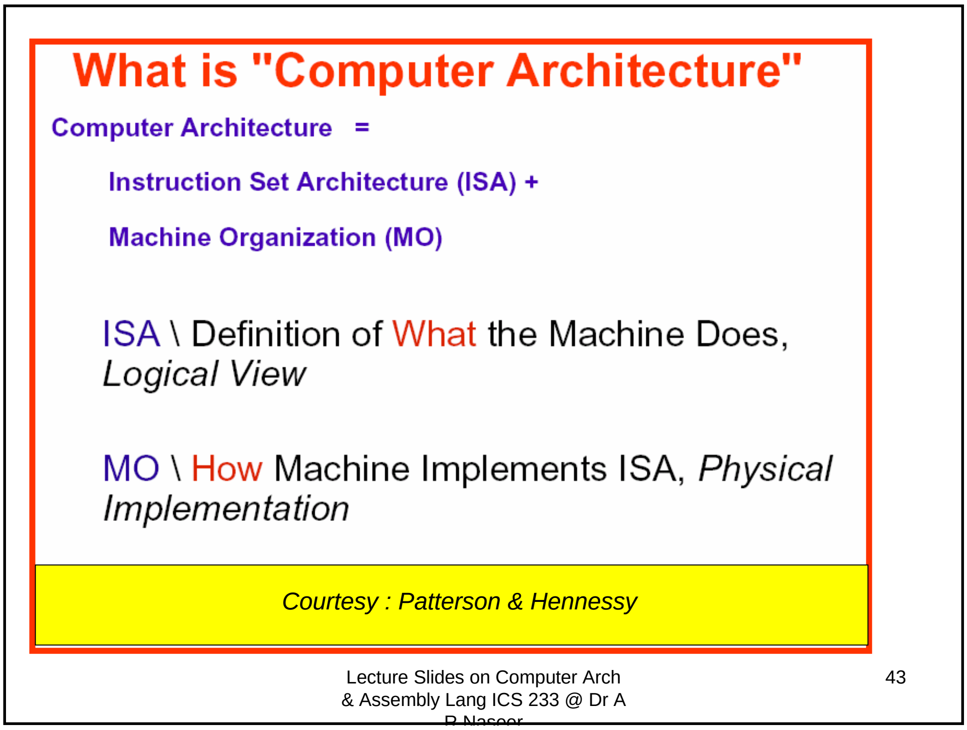
MARS Assembler and Simulator Tool


C:\Documents and Settings\Muhamed Mudawar\My Documents\ICS 233\Tools\MARS\Fibonacci.asm - MARS 3.2.1
File Edit Run Settings Tools Help
Edit Execute
Mars Messages Run I/O
Clear
Registers Coproc 1 Coproc 0

```
# Compute first twelve Fibonacci numbers and put in array, then print
        .data
fibs: .word   0 : 12        # "array" of 12 words to contain fib values
size: .word  12             # size of "array"
        .text
        la   $t0, fibs      # load address of array
        la   $t5, size      # load address of size variable
        lw   $t5, 0($t5)    # load array size
        li   $t2, 1         # 1 is first and second Fib. number
        add.d $f0, $f2, $f4
        sw   $t2, 0($t0)    # F[0] = 1
        sw   $t2, 4($t0)    # F[1] = F[0] = 1
        addi $t1, $t5, -2   # Counter for loop, will execute (size-2) times
loop: lw   $t3, 0($t0)      # Get value from array F[n]
        lw   $t4, 4($t0)    # Get value from array F[n+1]
        add  $t2, $t3, $t4  # $t2 = F[n] + F[n+1]
```

Line: 1 Column: 1 ☑ Show Line Numbers

Name	Number	Value
$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x00000000
$v1	3	0x00000000
$a0	4	0x00000000
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000000
$t1	9	0x00000000
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000



What is "Computer Architecture"

Computer Architecture =

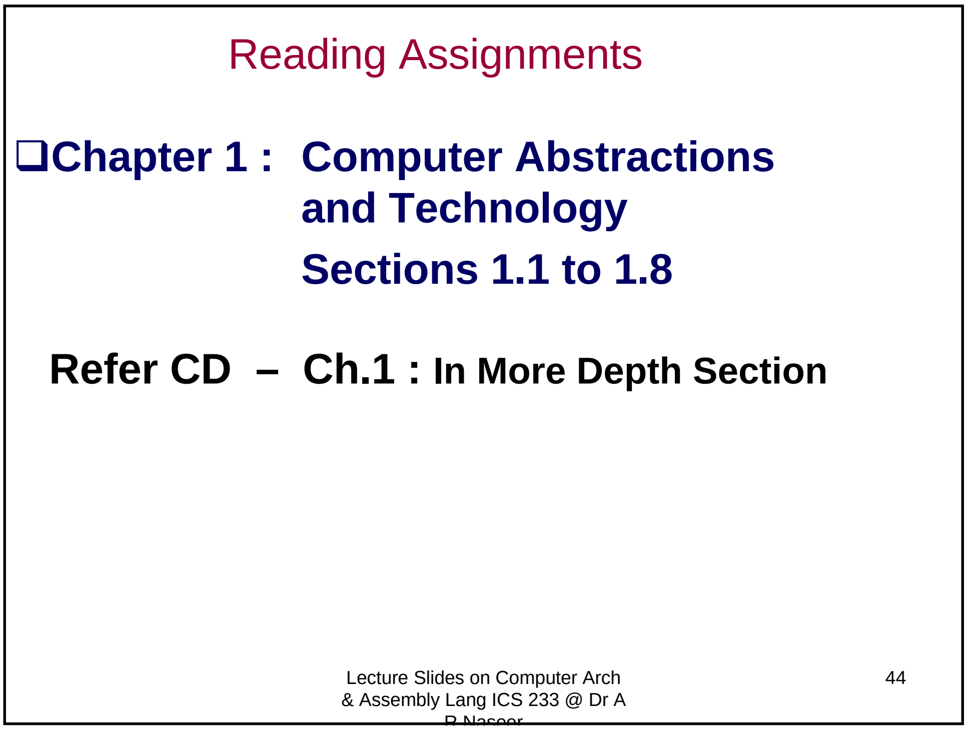
Instruction Set Architecture (ISA) +

Machine Organization (MO)

ISA \ Definition of What the Machine Does, *Logical View*

MO \ How Machine Implements ISA, *Physical Implementation*

Courtesy : Patterson & Hennessy



Reading Assignments

- ❑**Chapter 1 : Computer Abstractions and Technology**

  Sections 1.1 to 1.8

Refer CD – Ch.1 : In More Depth Section