Assembly Language
組合語言-資訊工程二乙
Lecture slides(2018 – 2019)

Fu Jen Catholic University, Dept of Computer Science and Information Engineering –CSIE

資訊工程系-輔仁大學

教授: 周賜福

107學年度第一學期

Assembly Language for x86 Processors

7th Edition

Kip Irvine

Chapter 2: x86 Processor Architecture

Slides prepared by the author and added some materials by the Instructor

Revision date: 2/15/2015

Chapter Overview

- **General Concepts**
- IA-32 Processor Architecture
- IA-32 Memory Management
- 64-bit Processor (New)
- Components of an IA-32 Microcomputer
- Input-Output System

General Concepts

- Basic microcomputer design
- Instruction execution cycle
- Reading from memory
- How programs run

We can divide the parts that make up a computer into three broad categories or subsystem: the **central processing unit (CPU)**, the **main memory** and the **input/output subsystem**.

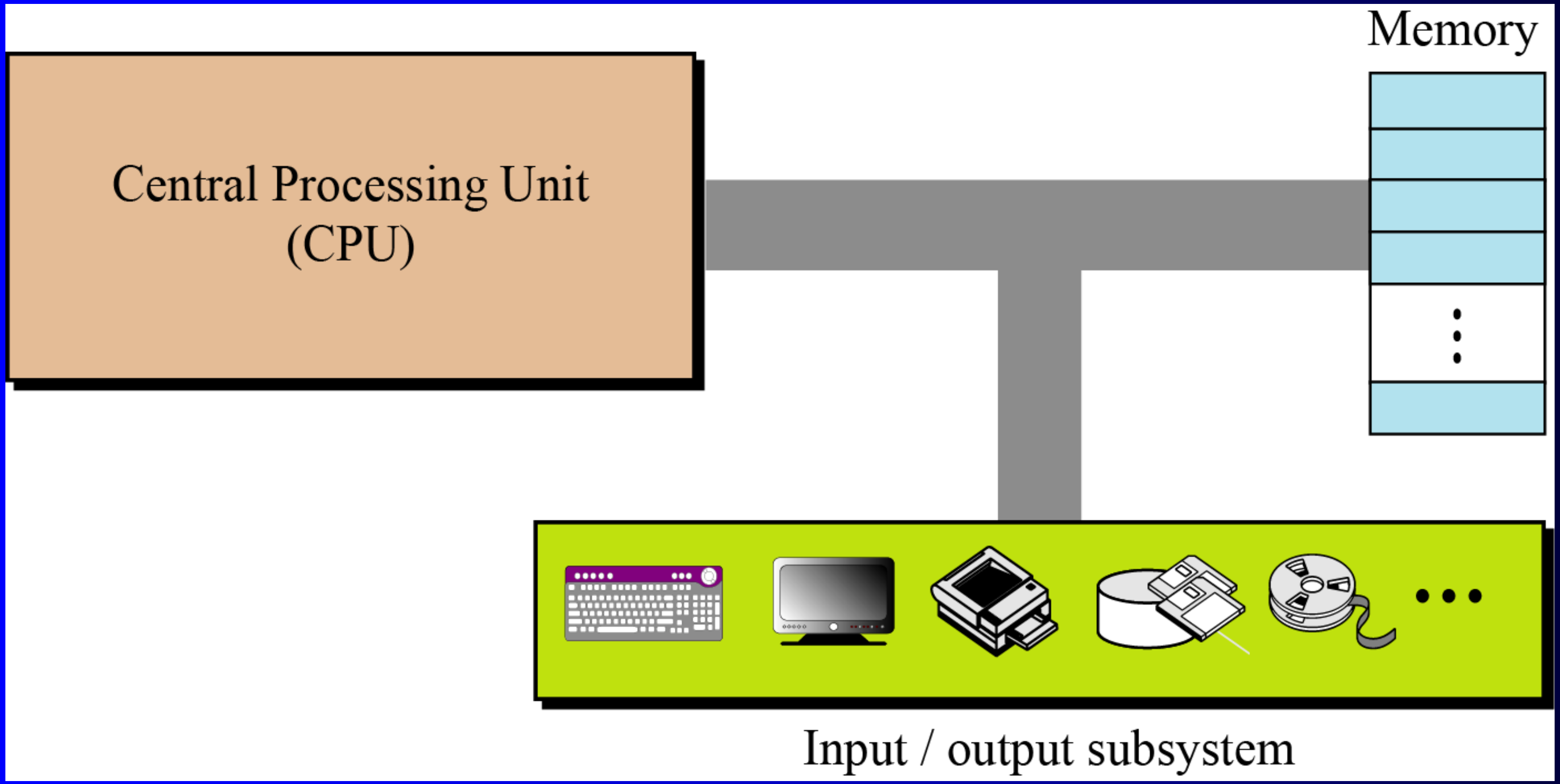
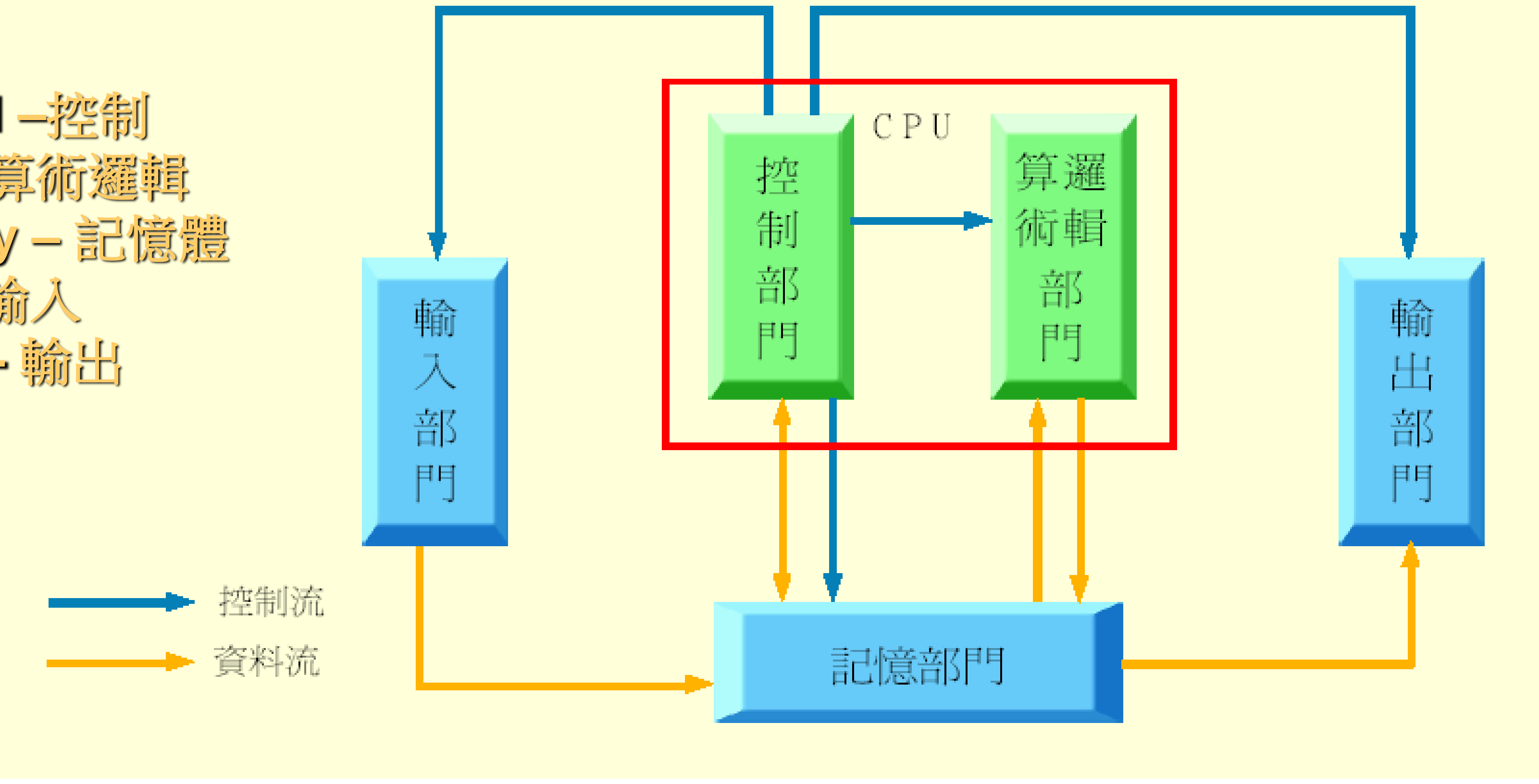


Figure Computer hardware (subsystems)

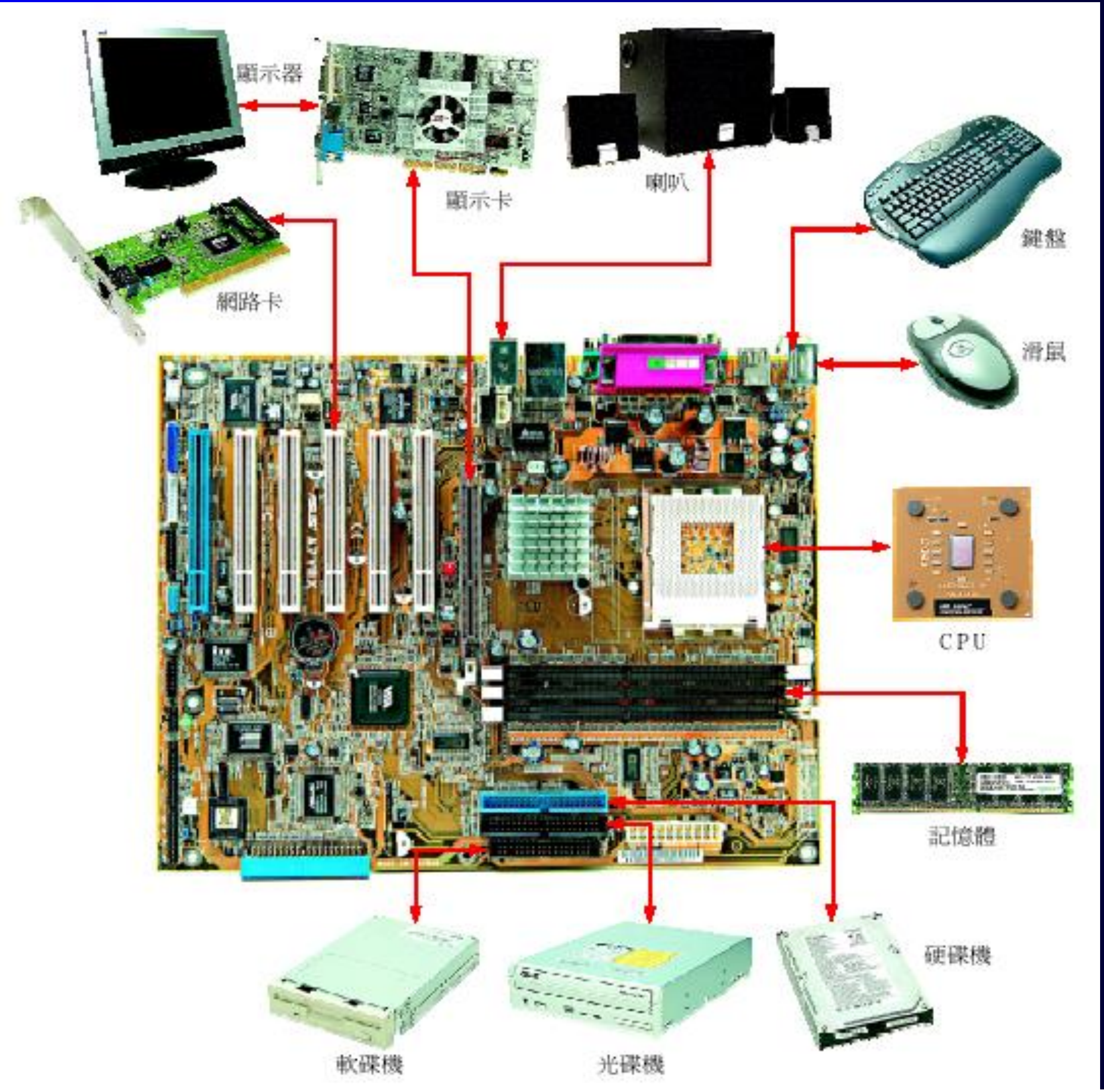
Computer hardware (subsystems)

Control –控制
ALU – 算術邏輯
Memory – 記憶體
Input- 輸入
Output- 輸出



圖表 2-13　硬體的五大部門

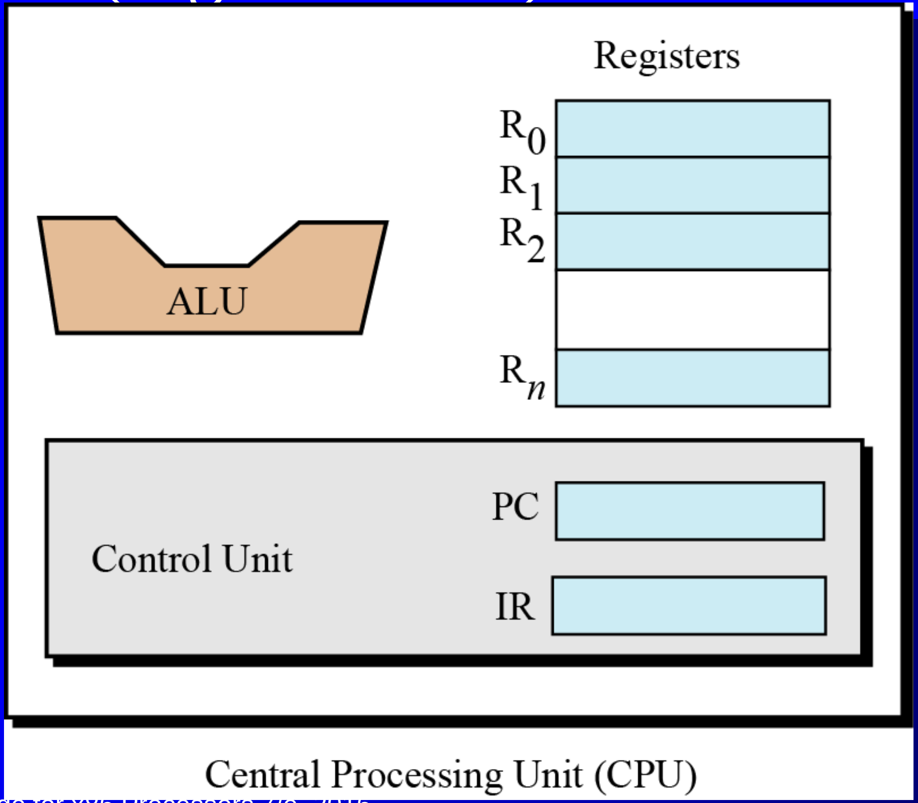
主機板



圖表 2-18 主機板

The arithmetic logic unit (ALU)

The central processing unit (CPU) performs operations on data. In most architectures it has three parts: an arithmetic logic unit (ALU), a control unit and a set of registers, fast storage locations (Figure below).



Central processing unit (CPU)

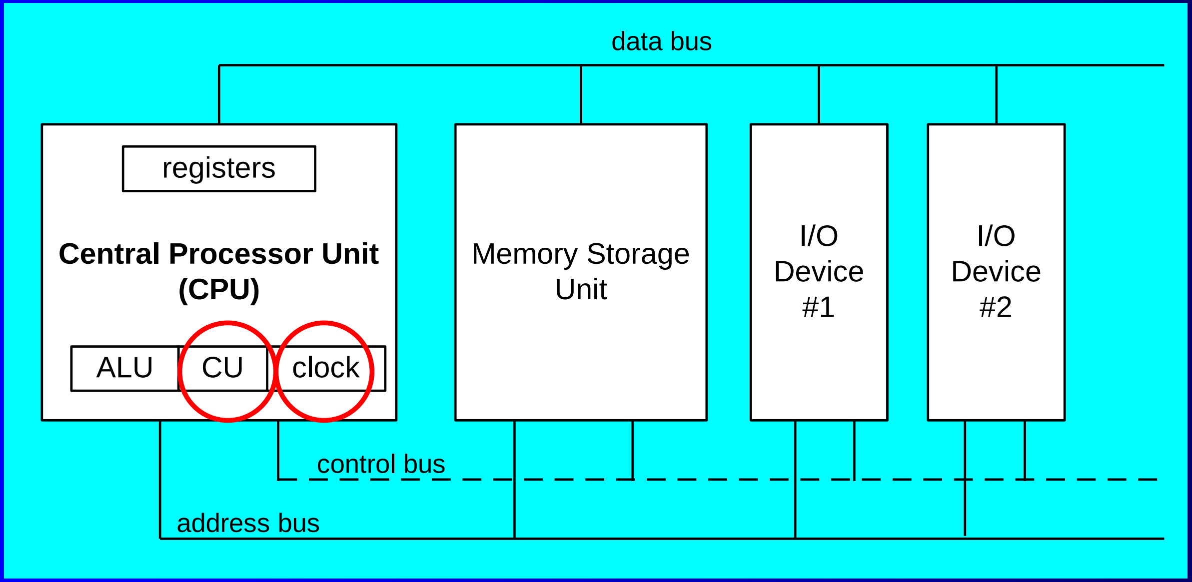
Registers

Registers are fast stand-alone storage locations that hold data temporarily. Multiple registers are needed to facilitate the operation of the CPU. Some of these registers are shown in Figure on the previous slide.

- ❑ Data registers
- ❑ Instruction register
- ❑ Program counter

Basic Microcomputer Design

- clock synchronizes CPU operations
- control unit (CU) coordinates sequence of execution steps
- ALU performs arithmetic and bitwise processing



Irvine, Kip R. Assembly Language for x86 Processors 7/e, 2016.

Microcomputer Design

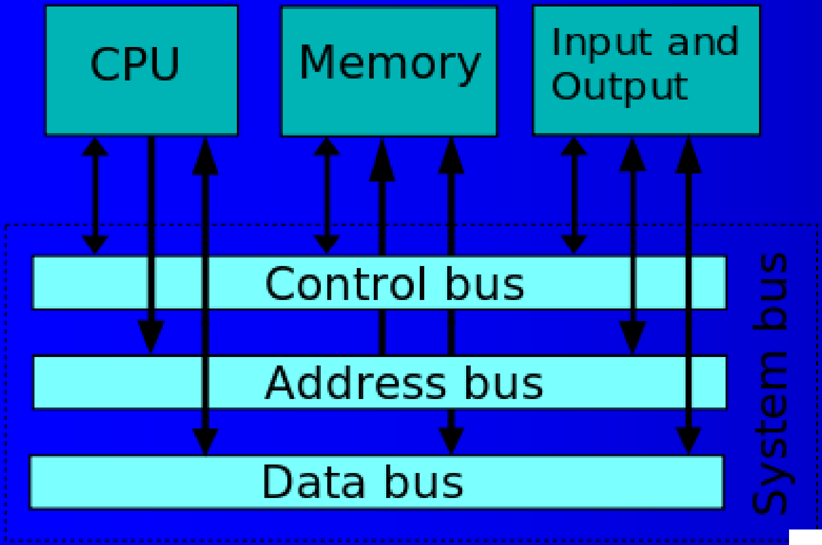
The CPU is attached to the rest of the computer via pins attached to the CPU socket in the computer's mother board.

Most pins connect to the data bus, the control bus, and the address bus.
The memory storage unit is where instructions and data are held while a computer program is running.

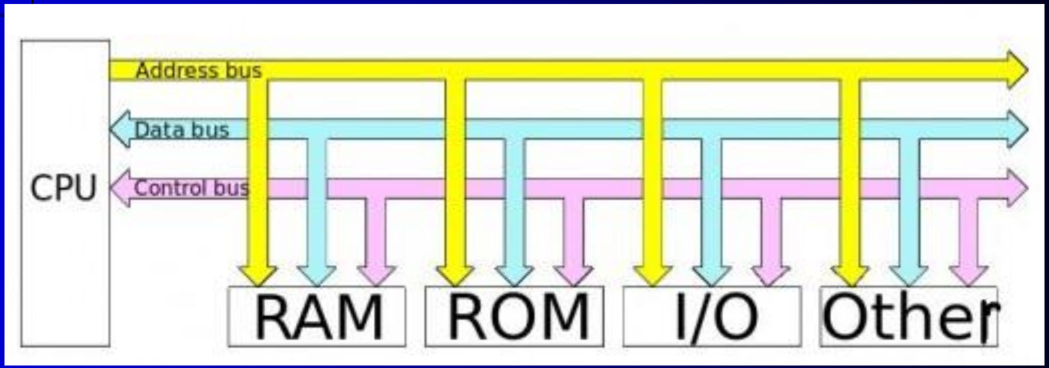
The storage unit receives requests for data from the CPU, transfers data from CPU into memory. All processing of data takes place within the CPU, so programs residing in memory must be copied into the CPU before they can execute. Individual program instructions can be copied into the CPU one at a time, or groups of instructions can be copied together.

What is a bus?

A bus is a group of parallel wires that transfer data from one part of the computer to another.



Depicting the Three different kinds of buses.



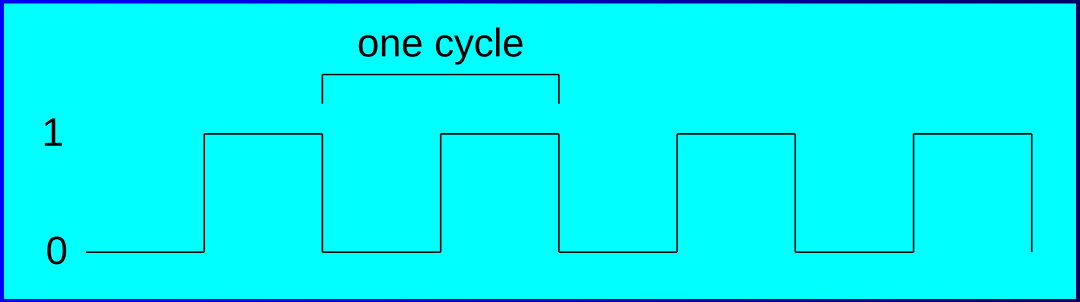
What is a bus?

A bus is a group of parallel wires that transfer data from one part of the computer to another.

A computer system usually contains four bus types: data, I/O, control, address.
The data bus transfers instructions and data between the CPU and memory.
The I/O bus transfers data between CPU and the system input/output devices.
The control bus uses binary signals to synchronize actions of all devices attached to the system bus.
The address bus holds the addresses of instructions and data when the currently executing instruction transfers data between the CPU and memory.

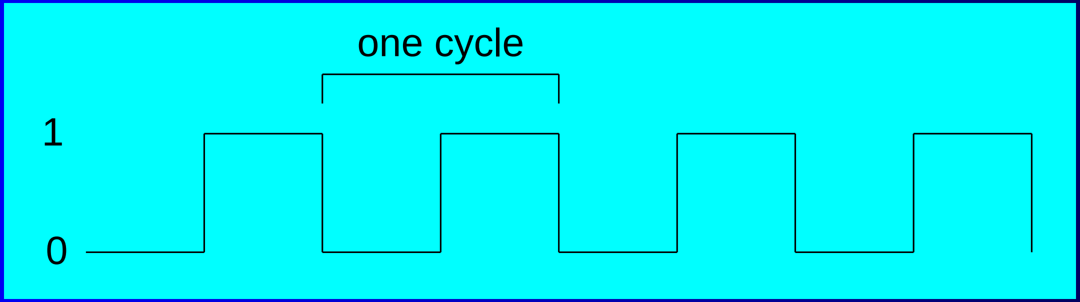
Clock

- synchronizes all CPU and BUS operations
- machine (clock) cycle measures time of a single operation
- clock is used to trigger events



Clock

- The duration of a clock cycle is calculated as the reciprocal of the clock's speed. Which in turn is measured in oscillation per second. A clock that oscillates 1 billion times per second (1GHz), for example produces a clock cycle with a duration of one billionth of a second (1 nanosecond).
- A machine instruction requires at least one clock cycle to execute, and a few require in excess of 50 clocks (the multiply instruction on the 8088 processor, for example).
- Instructions requiring memory access often have empty clock cycle called wait states because of the differences in the speeds of CPU, the system bus, and memory circuits.



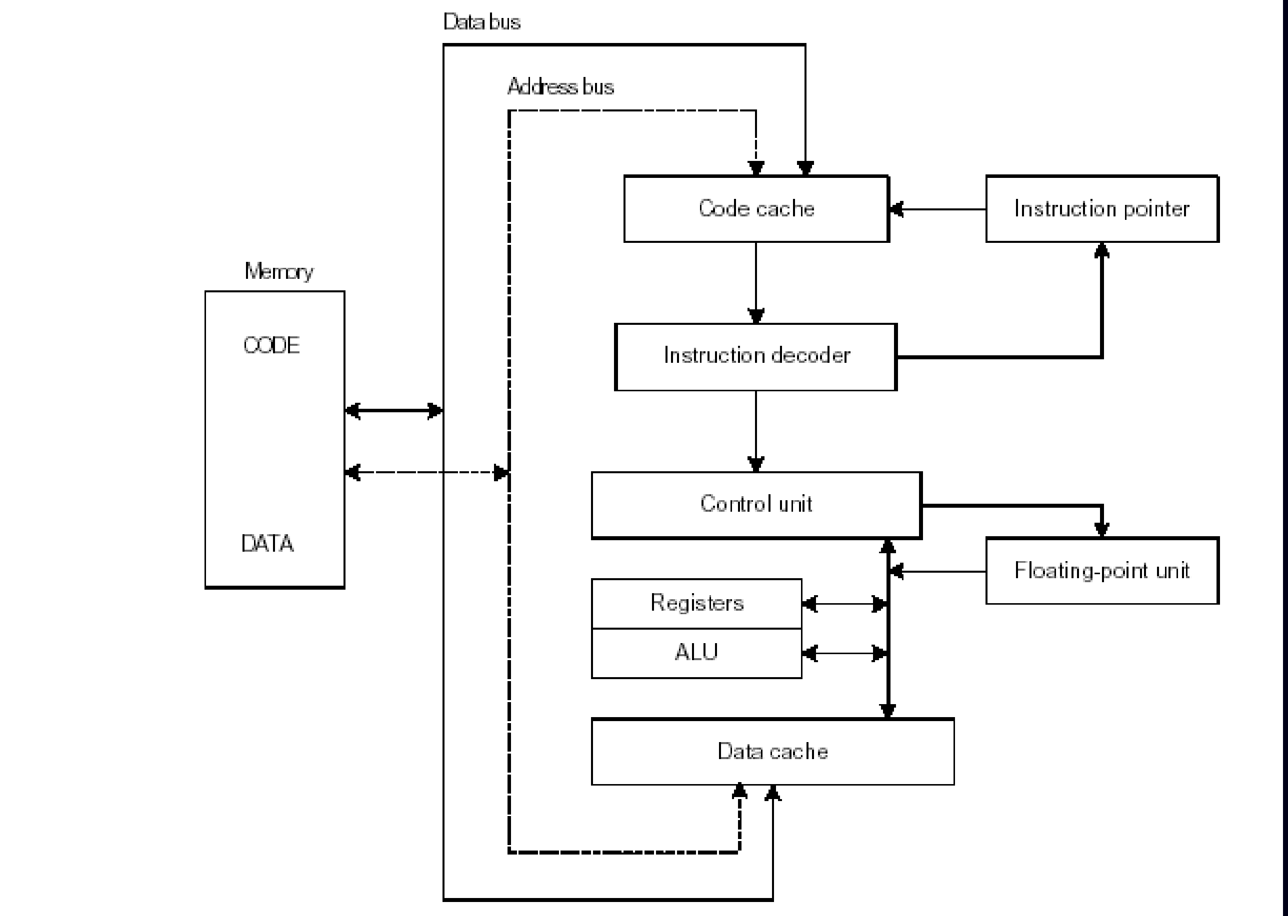
What's Next

- General Concepts
- IA-32 Processor Architecture
- IA-32 Memory Management
- 64-bit Processor
- Components of an IA-32 Microcomputer
- Input-Output System

Instruction Execution Cycle

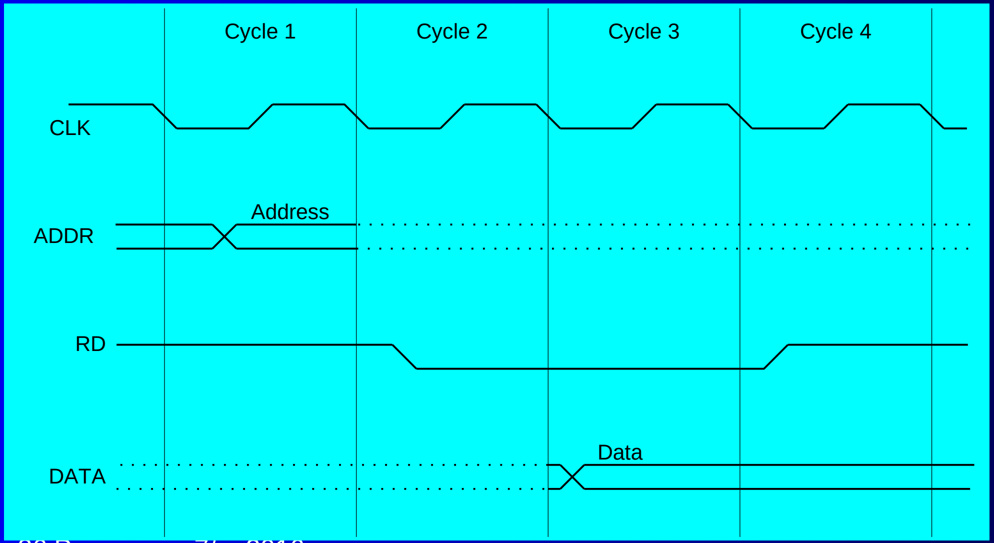
- Fetch
- Decode
- Fetch operands
- Execute
- Store output

Figure 2–2 Simplified Pentium CPU Block Diagram.



Reading from Memory

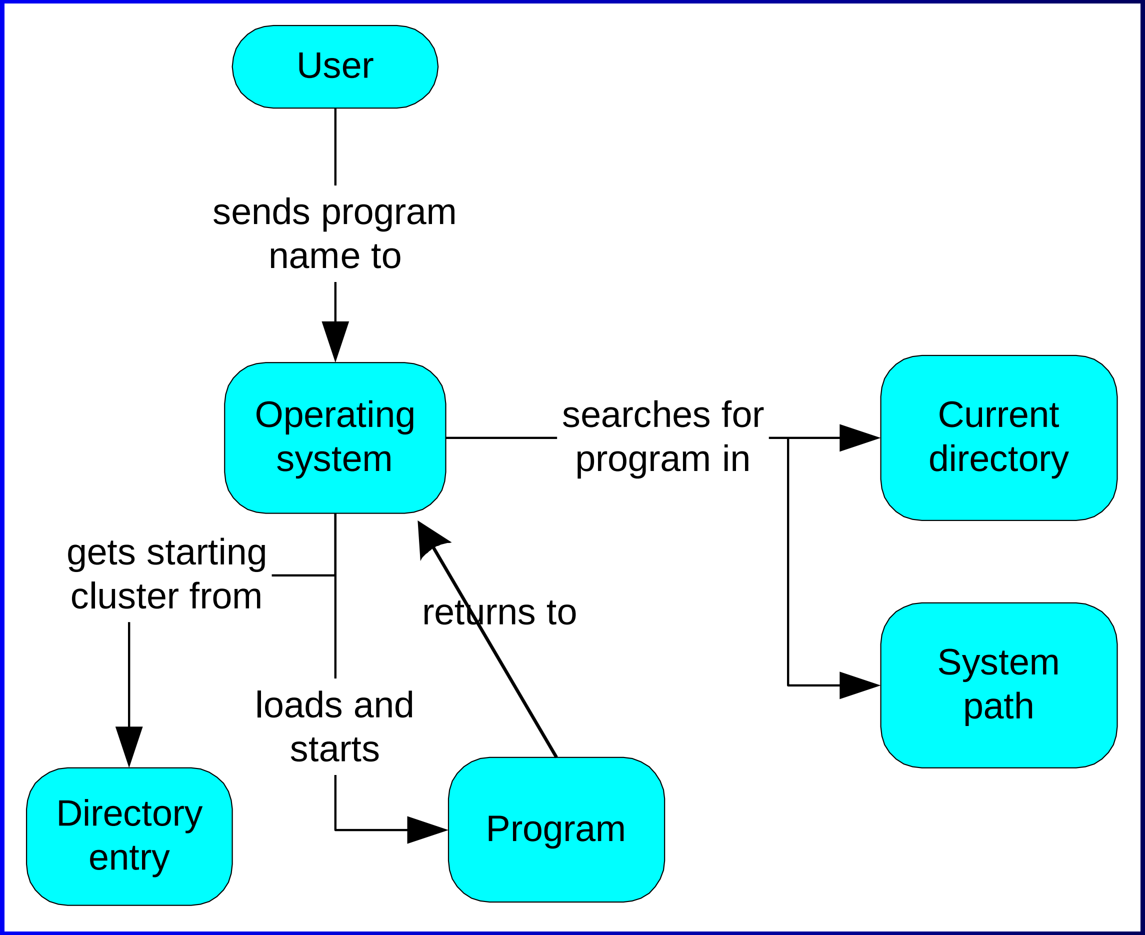
- Multiple machine cycles are required when reading from memory, because it responds much more slowly than the CPU. The steps are:
  - address placed on address bus
  - Read Line (RD) set low
  - CPU waits one cycle for memory to respond
  - Read Line (RD) goes to 1, indicating that the data is on the data bus



Cache Memory

- High-speed expensive static RAM both inside and outside the CPU.
  - Level-1 cache: inside the CPU or outside
  - Level-2 cache: outside the CPU
  - Level-3 cache: outside the CPU
- Cache hit: when data to be read is already in cache memory
- Cache miss: when data to be read is not in cache memory.

How a Program Runs



Multitasking

- OS can run multiple programs at the same time.
- Multiple threads of execution within the same program.
- Scheduler utility assigns a given amount of CPU time to each running program.
- Rapid switching of tasks
  - gives illusion that all programs are running at once
  - the processor must support task switching.

2.2 IA-32 Processor Architecture

- Modes of operation
- Basic execution environment
- Floating-point unit
- Intel Microprocessor history

Modes of Operation

- Protected mode
  - native mode (Windows, Linux)
- Real-address mode
  - native MS-DOS
- System management mode
  - power management, system security, diagnostics

- Virtual-8086 mode
  - hybrid of Protected
  - each program has its own 8086 computer

Modes of Operation

- **Protected mode**
  - Protected mode is the native state of the processor, in which all instructions and features are available. Programs are given separate memory areas named segments, and the processor prevents programs from referencing memory outside their assigned segments.
- **Real-address mode**
  - Real address mode implements the programming environment of an early Intel processor with a few extra features, such as the ability to switch into other modes. This mode is useful if a program requires direct access to system memory and hardware devices.
- **System management mode**
  - System management mode (SMM) provides an operating system with a mechanism for implementing functions such as power management and system security. These functions are usually implemented by computer manufacturers who customize the processor for a particular system setup.

Basic Execution Environment

- Addressable memory
- General-purpose registers
- Index and base registers
- Specialized register uses
- Status flags
- Floating-point, MMX, XMM registers

Addressable Memory

- Protected mode
  - 4 GB (Extended Physical Addressing allows a total of 64 GBytes of physical memory to be addressed.)
  - 32-bit address
- Real-address and Virtual-8086 modes
  - 1 MB space
  - 20-bit address

Let us Learn about Registers.

Very Important to write Assembly Language Program.

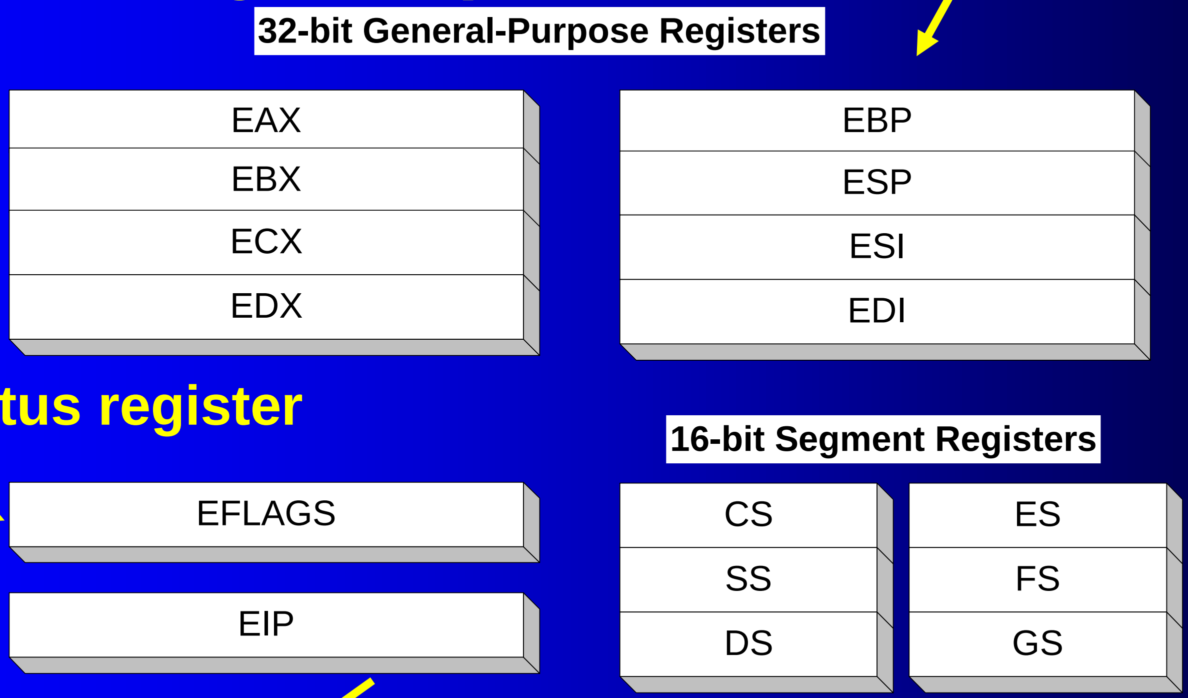
Intel 32 bit and 64 bit General purpose Registers

32-bit	64-bit
EAX	RAX
EBX	RBX
ECX	RCX
EDX	RDX
ESI	RSI
EDI	RDI
EBP	RBP
ESP	RSP
	R8
	R9
	R10
	R11
	R12
	R13
	R14
	R15

Irvine, Kip R. Assembly Language for x86 Processors 7/e, 2016.

General-Purpose Registers

Named storage locations inside the CPU, optimized for speed. (8 General purpose registers)



Process status register

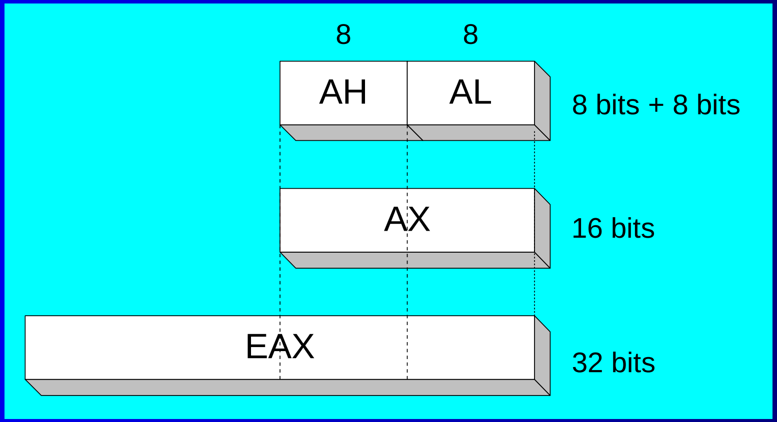
Instruction pointer

Six segment registers

Irvine, Kip R. Assembly Language for x86 Processors 7/e, 2016.

Accessing Parts of Registers

- Use 8-bit name, 16-bit name, or 32-bit name
- Applies to EAX, EBX, ECX, and EDX



Lower 16 bit is referenced as AX

32-bit	16-bit	8-bit (high)	8-bit (low)
EAX	AX	AH	AL
EBX	BX	BH	BL
ECX	CX	CH	CL
EDX	DX	DH	DL

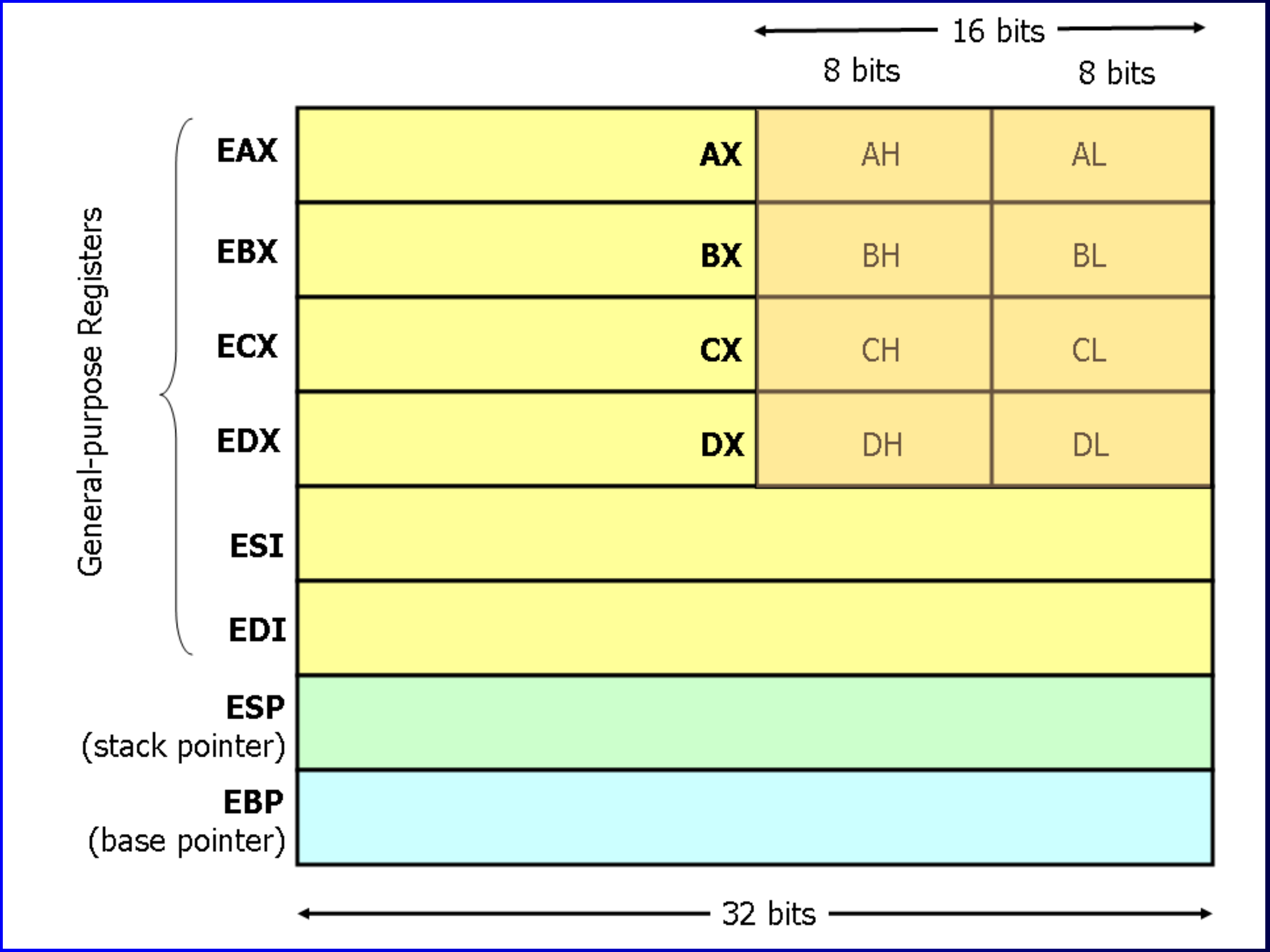
Index and Base Registers

- **Some registers have only a 16-bit name for their lower half:**

32-bit	16-bit
ESI	SI
EDI	DI
EBP	BP
ESP	SP

[More details on Intel architecture]

General-Purpose Registers



Irvine, Kip R. Assembly Language for x86 Processors 7/e, 2016.

Some Specialized Register Uses (1 of 2)

- General-Purpose
  - EAX – accumulator (multiplication and division)
  - ECX – loop counter (not used for arithmetic)
  - ESP – stack pointer (used for a system memory structure)
  - ESI, EDI – index registers (Extended source Index and Extended destination index)
  - EBP – extended frame pointer (stack)
- Segment
  - CS – code segment
  - DS – data segment
  - SS – stack segment
  - ES, FS, GS - additional segments

Assembly program Example

1. What you need to set up in order to run x86 Assembly language?

2. How to write assembly language and assemble the program?

3. Show you some simple assembly language program and run them.

4. You need to practice at home or other times in order to become familiar with the Assembly language program.

USING MASM Assembler and the link

公告日期 : 2015/9/22
附件 : 無
Assembly Language program Instructions for Assembling.

In order to run the Assembly programs that we will be learning in this course , please follow these instructions.

Please use the web link http://www.kipirvine.com

2. After entering the authors website, please choose "Getting started with MASM" link.
3. That page will explain clearly how to download the MASM for which visual studio version and run your first program. Try it yourself as to how to assemble the first assembly program and run them to see if your program works correctly.
4. The author clearly instructs you how to assemble the example program as well as write new programs and run them.

Good Luck and please try some assembly program before next class.

First Assembly program
Moving Integer Number into EAX

```
TITLE MASM Template    (1stassembly.asm)
; Description:     This is the first assembly program that adds
;                   two numbers.
; Author Name:     周賜福
; Author Number:   091111
; Revision date:   Sept 20th, 2016
INCLUDE Irvine32.inc  ; include library  for assembler
.code
main PROC
        mov  eax,  5      ; move 5 to the EAX register
        add   eax,  6      ; add 6 to the EAX register
        call   WriteInt    ; display value in EAX
        exit
main ENDP
END main
```

Second Assembly program
Moving Hexadecimal Number into EAX

```
TITLE MASM Template    (2ndassembly.asm)
; Description:     This is the 2nd assembly program that adds
;                  two hexadecimal numbers.
; Author Name:     周賜福
; Author Number:   091111
; Revision date:   Sept 20th, 2016
INCLUDE Irvine32.inc  ; include library  for assembler
.code
main PROC
    mov  eax,  10000h  ; Move a hex value into EAX
    add   eax,  40000h  ; add a hex value into EAX
    sub   eax,  20000h  ; subtract hex value from EAX
    call    DumpRegs    ; Display what is in Register

          exit
main ENDP
END main
```

Creating a Project From Scratch – Refer to the Author's website.

Visual Studio makes it possible (in 12 easy steps) to create an Assembly Language project from scratch.

In the first step, you will create a Win32 Console application designed for C++, and just modify the custom build rules.

Step 1: Select New from the File menu, then select Project. In the New Project window, select **Win32** under Visual C++ in the left panel, and select **Win32 Console Application** in the middle panel. Give your project a suitable name (near the bottom of the window):

For the rest, follow the author's Website and set your project

Try to write similar simple assembly programs by yourself!

Hint: How about moving -128 to EAX register.
Add + 54 and see if you get the result.

Similar ideas can be thought of by you.

Some Specialized Register Uses (2 of 2)

- **EIP – instruction pointer** (register contains the address of the next instruction to be executed)
- **EFLAGS**
  - **status and control flags**
  - **each flag is a single binary bit** (Reflect the outcome of some CPU operation.)

In the high level language program, you do not know if the overflow occurs or not.

In the assembly language program, you need to worry about the overflow of the program.

Status Flags

- Carry
  - unsigned arithmetic out of range
- Overflow
  - signed arithmetic out of range
- Sign
  - result is negative
- Zero
  - result is zero
- Auxiliary Carry
  - carry from bit 3 to bit 4
- Parity
  - sum of 1 bits is an even number

MMX Technology

What is MMX Register?

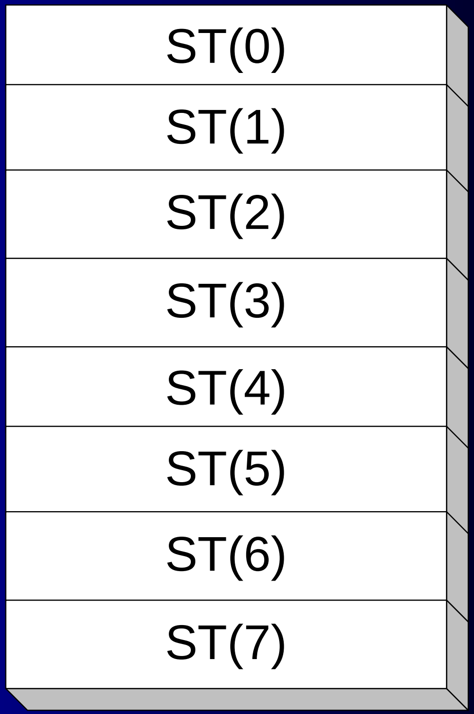
MMX technology was added onto the Pentium processor by Intel to improve the performance of advanced multimedia and communications applications. The eight 64-bit MMX registers support special instructions called SIMD (single instruction, Multiple-Data). As the name implies, MMX instructions operate in parallel on the data values contained in MMX registers.

What XMM Registers?

The x86 architecture also contains eight 128-bit registers called XMM registers. They are used by streaming SIMD extensions to the instruction set.

Floating-Point, MMX, XMM Registers

- Eight 80-bit floating-point data registers
  - ST(0), ST(1), . . . , ST(7)
  - arranged in a stack
  - used for all floating-point arithmetic
- Eight 64-bit MMX registers
- Eight 128-bit XMM registers for single-instruction multiple-data (SIMD) operations



MMX technology was added onto the Pentium processor by Intel to improve the performance of advanced multimedia and communications applications. The eight 64-bit MMX registers support special instructions called SIMD (single-Instruction and Multiple-Data)

HISTORY of INTEL Architecture

Intel Microprocessor History

- Intel 8086, 80286
- IA-32 processor family
- P6 processor family
- CISC and RISC

Early Intel Microprocessors

- **Intel 8080 (First processor)[8-bit]**
  - 64K addressable RAM
  - 8-bit registers
  - CP/M operating system
  - S-100 BUS architecture
  - 8-inch floppy disks!
- **Intel 8086/8088 (1978) [16-bit registers][1980 IBM had the identical 8088 processor]**
  - IBM-PC Used 8088
  - 1 MB addressable RAM
  - 16-bit registers
  - 16-bit data bus (8-bit for 8088)
  - separate floating-point unit (8087)

The IBM-AT

- Intel 80286 [some where between 1978 to 1985]
  - 16 MB addressable RAM
  - Protected memory
  - 24-bit address bus
  - several times faster than 8086
  - introduced IDE bus architecture
  - 80287 floating point unit

Intel IA-32 Family

- **Intel386 [1985]**
  - **4 GB addressable RAM, 32-bit registers, paging (virtual memory)-32 address bus Intel- (IA-32) family**
- **Intel486 [1989]**
  - **instruction pipelining, permitted multiple instructions to be processed at the same time.**
- **Pentium [1993]**
  - **superscalar, 32-bit address bus, 64-bit internal data path, MMX technology to the IA-32 family added.**

64-bit Processors

- **Intel64 [64-bit processing]**
  - 64-bit linear address space
  - Intel: Pentium Extreme, Xeon, Celeron D, Pendium D, Core 2, and Core i7
- **IA-32e Mode**
  - Compatibility mode for legacy 16- and 32-bit applications
  - 64-bit Mode uses 64-bit addresses and operands

Intel Technologies

- HyperThreading technology
  - two tasks execute on a single processor at the same time
- Dual Core processing
  - multiple processor cores in the same IC package
  - each processor has its own resources and communication path with the bus

Intel Processor Families

Currently Used:

- Pentium & Celeron – dual core
- Core 2 Duo - 2 processor cores
- Core 2 Quad - 4 processor cores
- Core i7 – 4 processor cores

CISC and RISC

- CISC – complex instruction set
  - large instruction set
  - high-level operations
  - requires microcode interpreter
  - examples: Intel 80x86 family
- RISC – reduced instruction set
  - simple, atomic instructions
  - small instruction set
  - directly executed by hardware
  - examples:
    - ARM (Advanced RISC Machines)
    - DEC Alpha (now Compaq)

What's Next

- General Concepts
- IA-32 Processor Architecture
- IA-32 Memory Management
- Components of an IA-32 Microcomputer
- Input-Output System

2.3 IA-32 Memory Management

- Real-address mode
- Calculating linear addresses
- Protected mode
- Multi-segment model
- Paging

Real-Address mode

- 1 MB RAM maximum addressable
- Application programs can access any area of memory
- Single tasking
- Supported by MS-DOS operating system
- Windows 95, 98 can be booted into this mode.

Protected Mode (1 of 2)

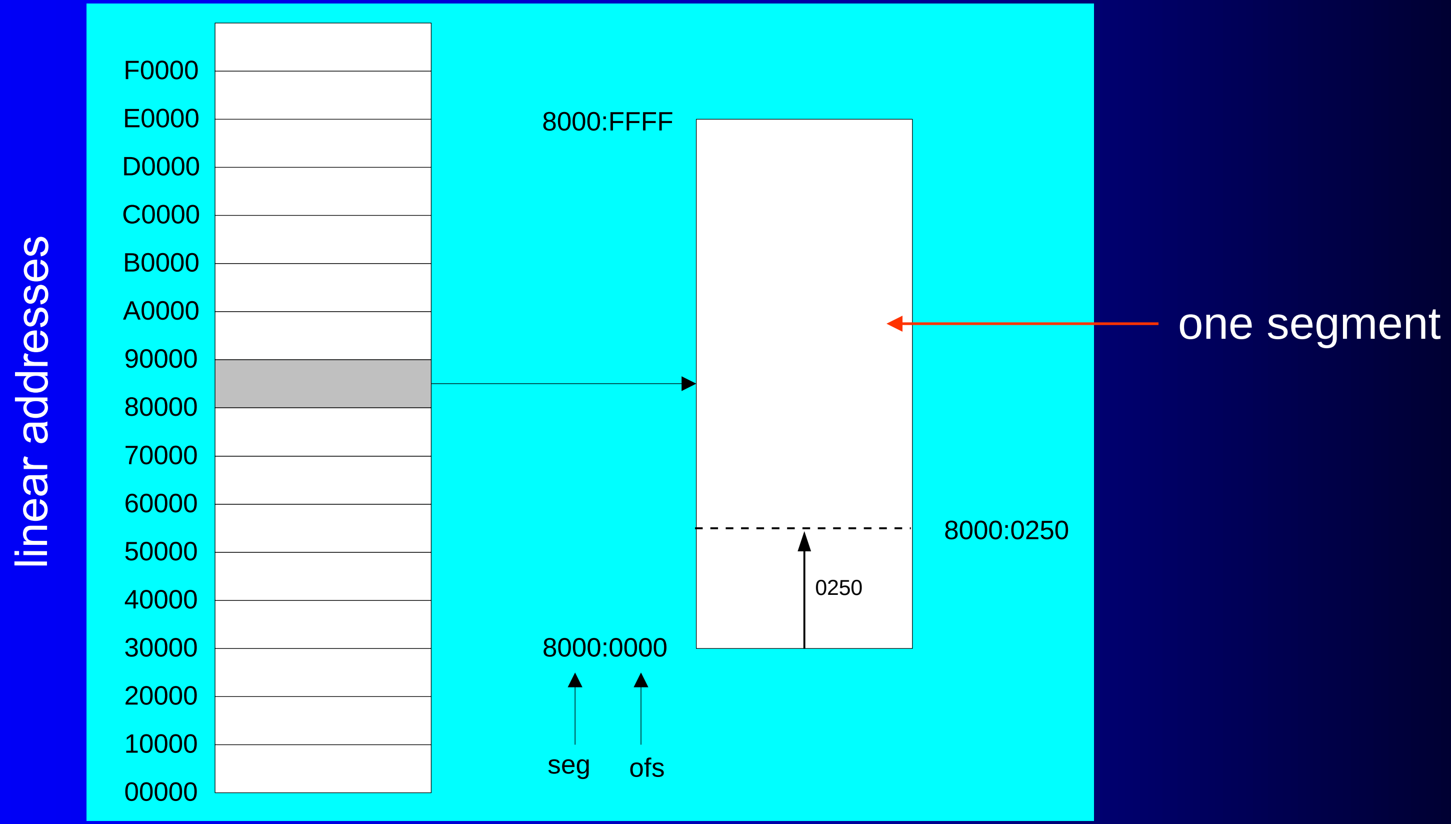
- 4 GB addressable RAM
  - (00000000 to FFFFFFFFh)
- Each program assigned a memory partition which is protected from other programs
- Designed for multitasking
- Supported by Linux & MS-Windows

Protected mode (2 of 2)

- Segment descriptor tables
- Program structure
  - code, data, and stack areas
  - CS, DS, SS segment descriptors
  - global descriptor table (GDT)
- MASM Programs use the Microsoft **flat** memory model

Segmented Memory

Segmented memory addressing: absolute (linear) address is a combination of a 16-bit segment value added to a 16-bit offset



Calculating Linear Addresses

- Given a segment address, multiply it by 16 (add a hexadecimal zero), and add it to the offset
- Example: convert 08F1:0100 to a linear address

```
Adjusted Segment value: 0 8 F 1 0
Add the offset:           0 1 0 0
Linear address:         0 9 0 1 0
```

Your turn . . .

What linear address corresponds to the segment/offset address 028F:0030?

028F0 + 0030 = 02920

Always use hexadecimal notation for addresses.

Your turn . . .

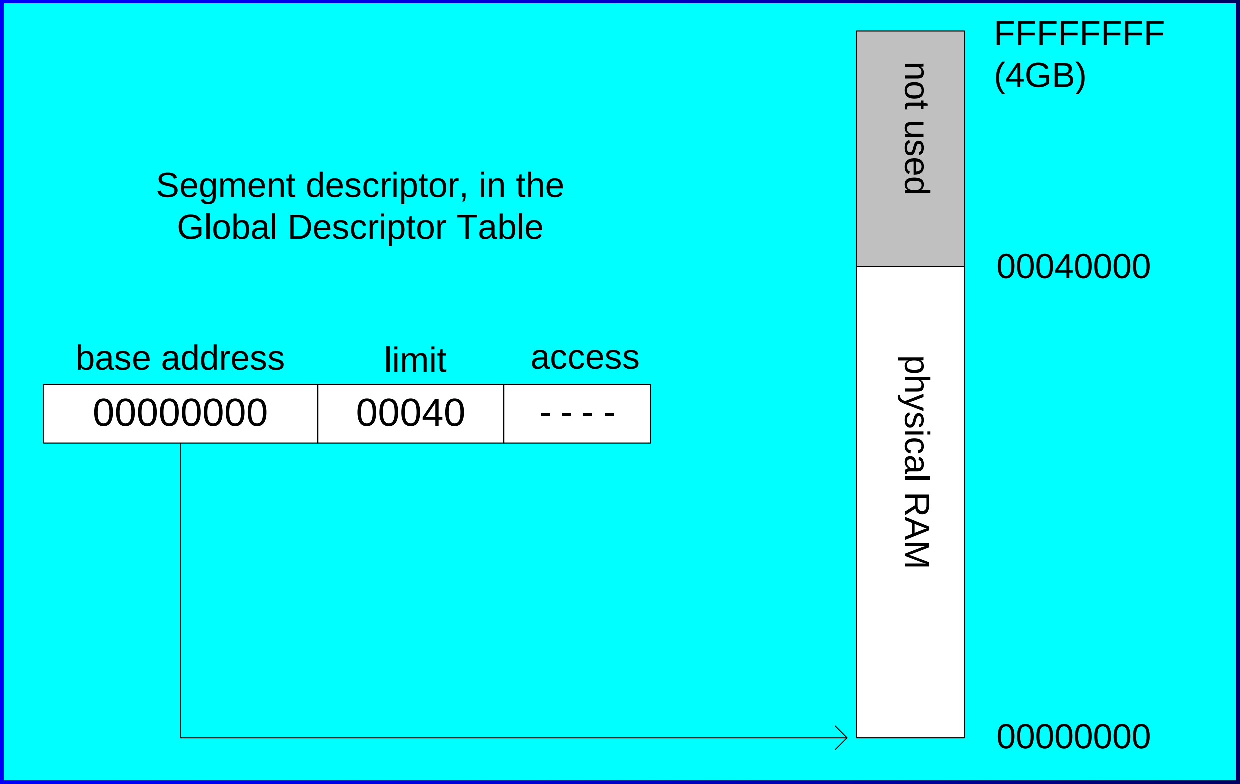
What segment addresses correspond to the linear address 28F30h?

Many different segment-offset addresses can produce the linear address 28F30h. For example:

28F0:0030, 28F3:0000, 28B0:0430, . . .

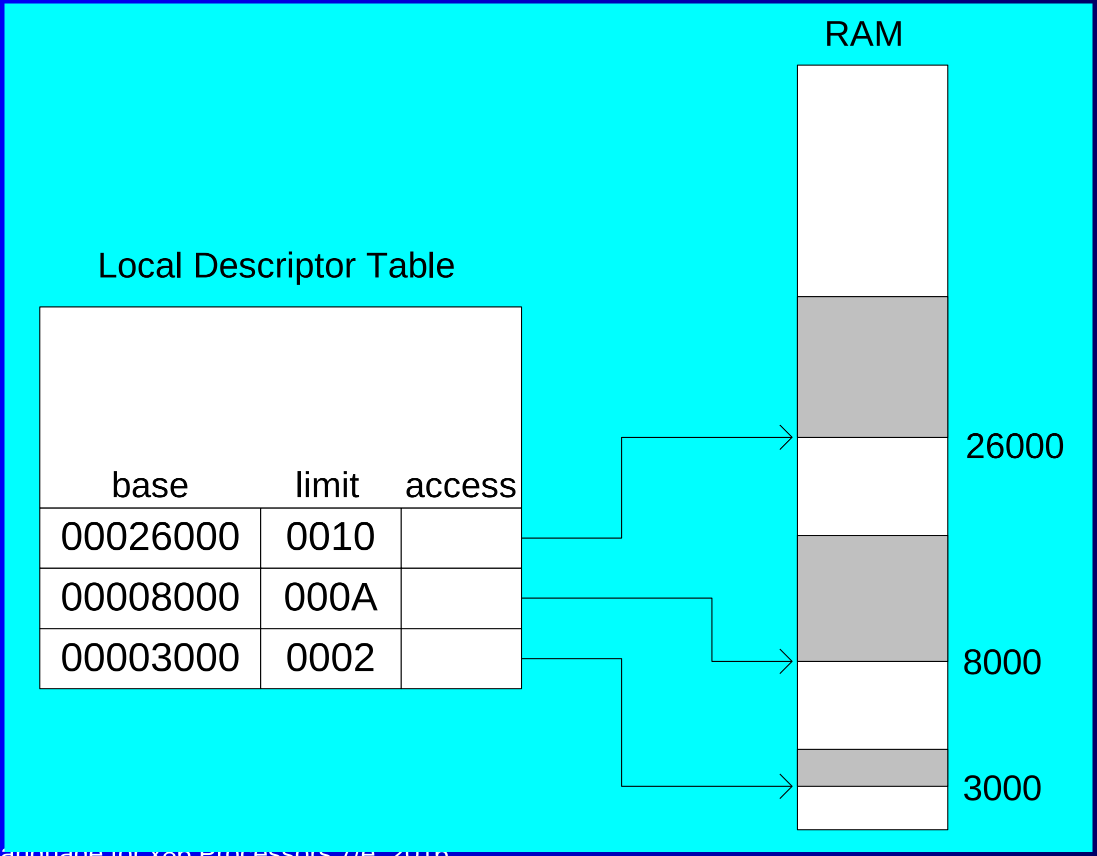
Flat Segment Model

- **Single global descriptor table (GDT).**
- **All segments mapped to entire 32-bit address space**



Multi-Segment Model

- **Each program has a local descriptor table (LDT)**
  - **holds descriptor for each segment used by the program**



Paging

- Supported directly by the CPU
- Divides each segment into 4096-byte blocks called **pages**
- Sum of all programs can be larger than physical memory
- Part of running program is in memory, part is on disk
- **Virtual memory manager** (VMM) – OS utility that manages the loading and unloading of pages
- **Page fault** – issued by CPU when a page must be loaded from disk

What's Next

- General Concepts
- IA-32 Processor Architecture
- IA-32 Memory Management
- 64-bit processor
- Components of an IA-32 Microcomputer
- Input-Output System

64-Bit Processors

- 64-Bit Operation Modes
  - Compatibility mode – can run existing 16-bit and 32-bit applications (Windows supports only 32-bit apps in this mode)
  - 64-bit mode – Windows 64 uses this
- Basic Execution Environment
  - addresses can be 64 bits (48 bits, in practice)
  - 16 64-bit general purpose registers
  - 64-bit instruction pointer named RIP

64-Bit General Purpose Registers

- 32-bit general purpose registers:
  - EAX, EBX, ECX, EDX, EDI, ESI, EBP, ESP, R8D, R9D, R10D, R11D, R12D, R13D, R14D, R15D
- 64-bit general purpose registers:
  - RAX, RBX, RCX, RDX, RDI, RSI, RBP, RSP, R8, R9, R10, R11, R12, R13, R14, R15

What's Next

- General Concepts
- IA-32 Processor Architecture
- IA-32 Memory Management
- 64-Bit Processors
- **Components of an IA-32 Microcomputer**
- Input-Output System

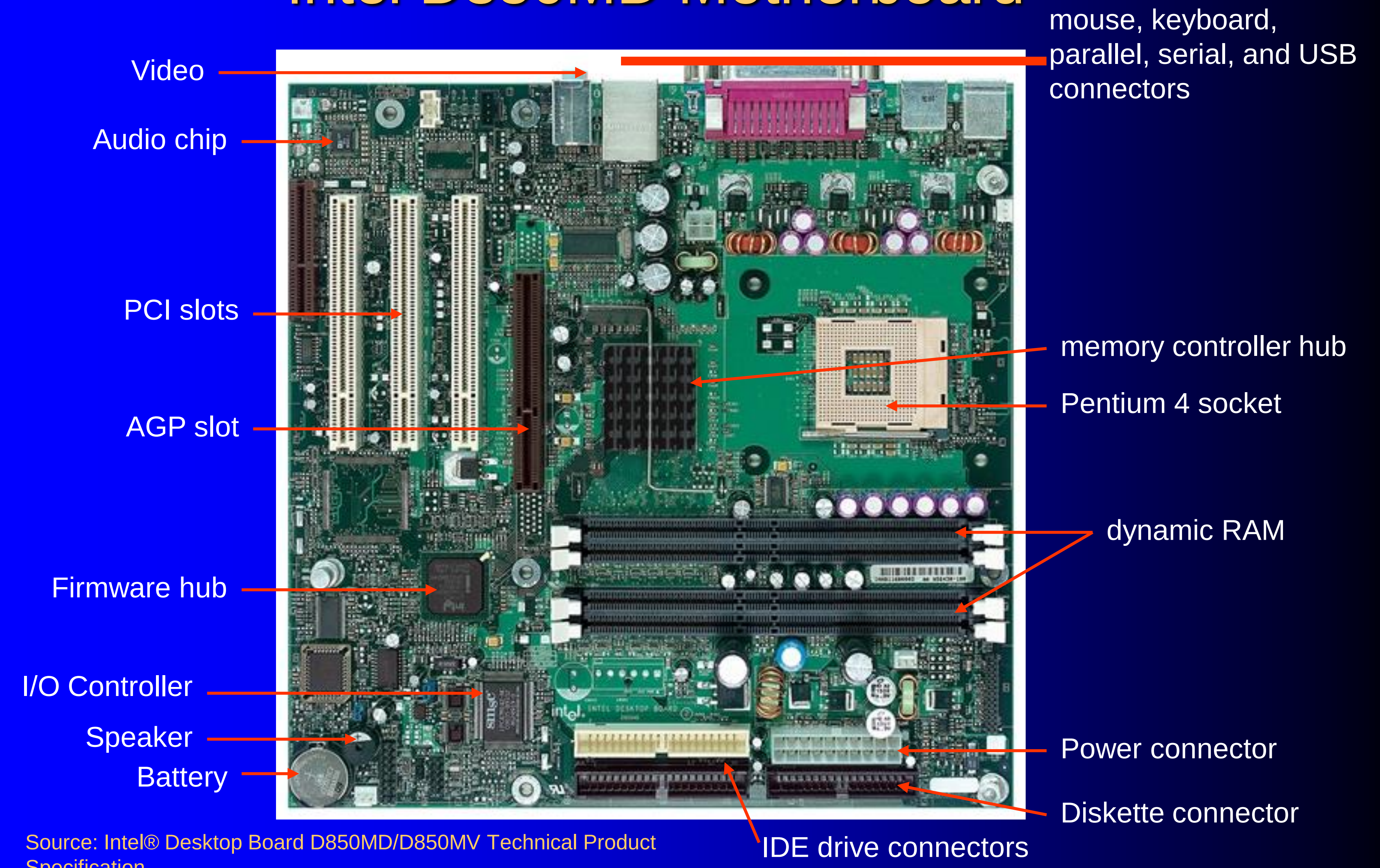
Components of an IA-32 Microcomputer

- Motherboard
- Video output
- Memory
- Input-output ports

Motherboard

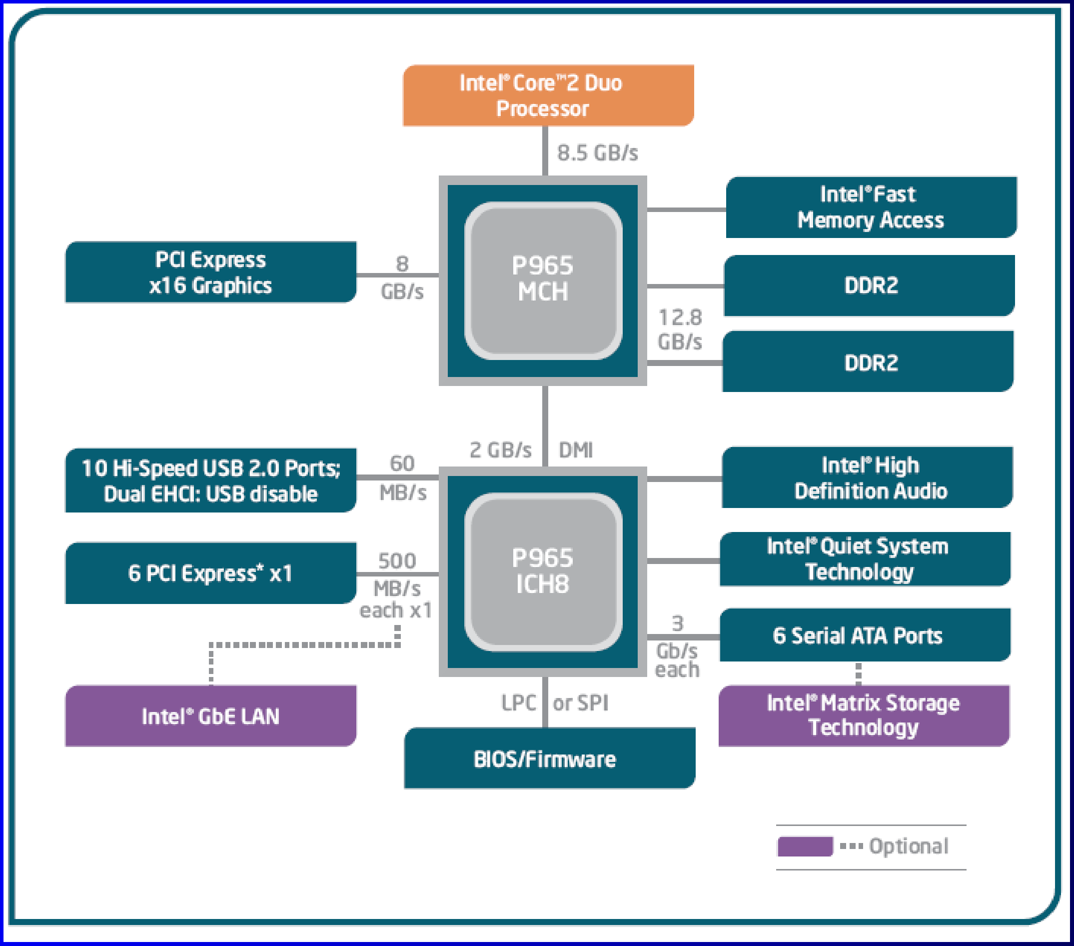
- CPU socket
- External cache memory slots
- Main memory slots
- BIOS chips
- Sound synthesizer chip (optional)
- Video controller chip (optional)
- IDE, parallel, serial, USB, video, keyboard, joystick, network, and mouse connectors
- PCI bus connectors (expansion cards)

Intel D850MD Motherboard



Source: Intel® Desktop Board D850MD/D850MV Technical Product Specification

Intel 965 Express Chipset



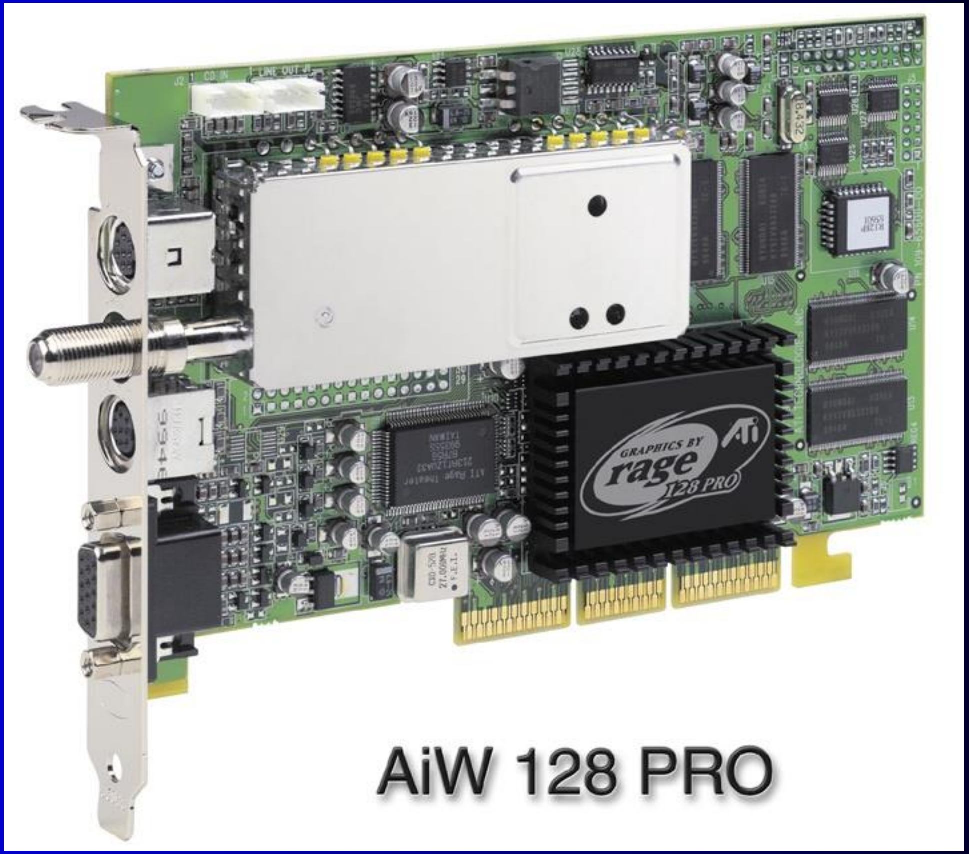
Video Output

- Video controller
  - on motherboard, or on expansion card
  - AGP (accelerated graphics port technology)*
- Video memory (VRAM)
- Video CRT Display
  - uses raster scanning
  - horizontal retrace
  - vertical retrace
- Direct digital LCD monitors
  - no raster scanning required

* This link may change over time.

Sample Video Controller (ATI Corp.)

- **128-bit 3D graphics performance powered by RAGE™ 128 PRO**
- **3D graphics performance**
- **Intelligent TV-Tuner with Digital VCR**
- **TV-ON-DEMAND™**
- **Interactive Program Guide**
- **Still image and MPEG-2 motion video capture**
- **Video editing**
- **Hardware DVD video playback**
- **Video output to TV or VCR**



Memory

- ROM
  - read-only memory
- EPROM
  - erasable programmable read-only memory
- Dynamic RAM (DRAM)
  - inexpensive; must be refreshed constantly
- Static RAM (SRAM)
  - expensive; used for cache memory; no refresh required
- Video RAM (VRAM)
  - dual ported; optimized for constant video refresh
- CMOS RAM
  - complimentary metal-oxide semiconductor
  - system setup information
- See: Intel platform memory (Intel technology brief: link address may change)

Input-Output Ports

- USB (universal serial bus)
  - intelligent high-speed connection to devices
  - up to 12 megabits/second
  - USB hub connects multiple devices
  - *enumeration*: computer queries devices
  - supports *hot* connections
- Parallel
  - short cable, high speed
  - common for printers
  - bidirectional, parallel data transfer
  - Intel 8255 controller chip

Input-Output Ports (cont)

- Serial
  - RS-232 serial port
  - one bit at a time
  - uses long cables and modems
  - 16550 UART (universal asynchronous receiver transmitter)
  - programmable in assembly language

Device Interfaces

- ATA host adapters
  - intelligent drive electronics (hard drive, CDROM)
- SATA (Serial ATA)
  - inexpensive, fast, bidirectional
- FireWire
  - high speed (800 MB/sec), many devices at once
- Bluetooth
  - small amounts of data, short distances, low power usage
- Wi-Fi (wireless Ethernet)
  - IEEE 802.11 standard, faster than Bluetooth

What's Next

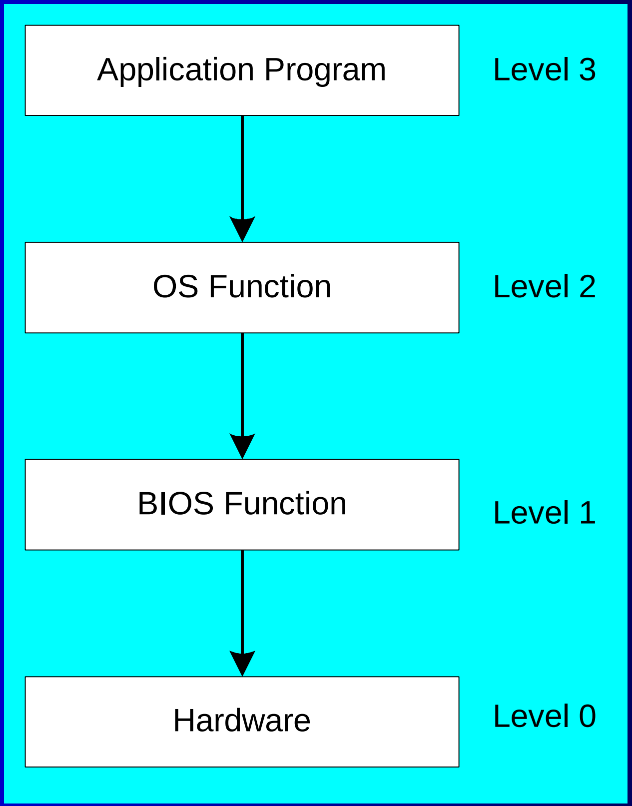
- General Concepts
- IA-32 Processor Architecture
- IA-32 Memory Management
- Components of an IA-32 Microcomputer
- Input-Output System

Levels of Input-Output

- Level 3: High-level language function
  - examples: C++, Java
  - portable, convenient, not always the fastest
- Level 2: Operating system
  - Application Programming Interface (API)
  - extended capabilities, lots of details to master
- Level 1: BIOS
  - drivers that communicate directly with devices
  - OS security may prevent application-level code from working at this level

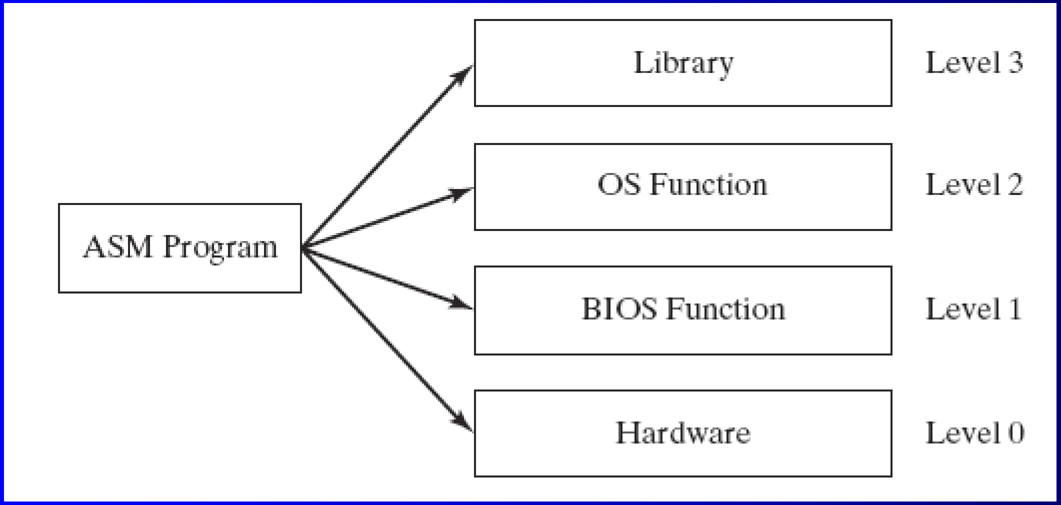
Displaying a String of Characters

When a HLL program displays a string of characters, the following steps take place:



Programming levels

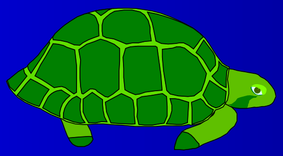
Assembly language programs can perform input-output at each of the following levels:



Irvine, Kip R. Assembly Language for x86 Processors 7/e, 2016.

Summary

- Central Processing Unit (CPU)
- Arithmetic Logic Unit (ALU)
- Instruction execution cycle
- Multitasking
- Floating Point Unit (FPU)
- Complex Instruction Set
- Real mode and Protected mode
- Motherboard components
- Memory types
- Input/Output and access levels



42 69 6E 61 72 79

What does this say?