X86 MICROPROCESSOR ARCHITECTURE

MODULE OUTLINE

1. GENARAL CONCEPTS
2. X86 ARCHITECTURE DETAILS
3. X86 MEMORY MANAGEMENT
4. COMPONENTS OF A TYPICAL X86 COMPUTER
5. INPUT-OUTPUT SYSTEM

2.1. GENERAL CONCCEPTS-1

2.1.1. INTRODUCTION

- **X86 family of processors include:**
  - **All Intel IA-32 processors: Pentium, Core-Duo;**
  - **AMD (Advanced Micro Devices): Athlon, Phenom, and Opteron**
- **Knowledge of hardware – good for Assembly programming**
- **Not exhaustive treatment of x86 processors**

2.1. GENERAL CONCCEPTS-2

2.1.2. BASIC MICROCOMPTER DESIGN

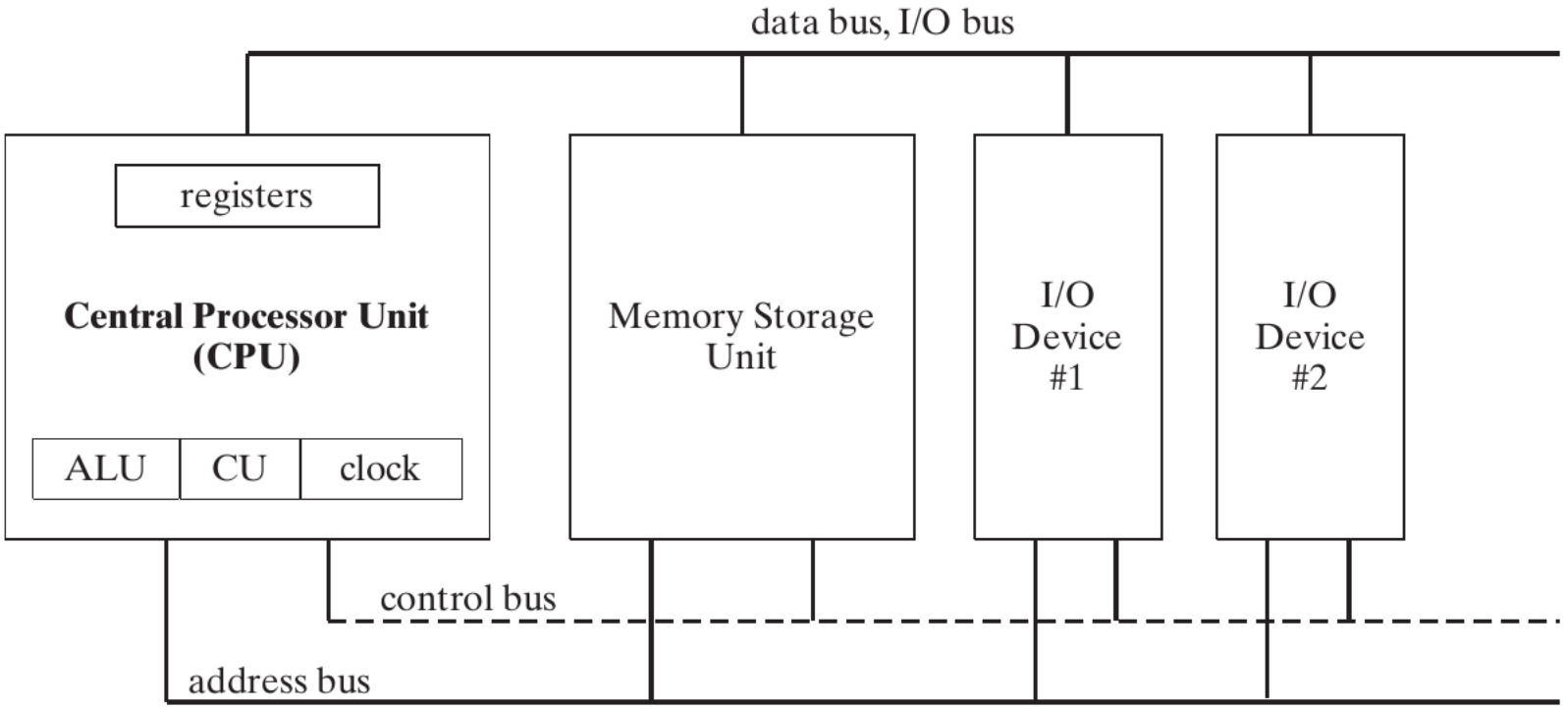


Figure 2.1.2.1 Block Diagram of a Microcomputer:

2.1. GENERAL CONCCEPTS-3

2.1.2. BASIC MICROCOMPTER DESIGN

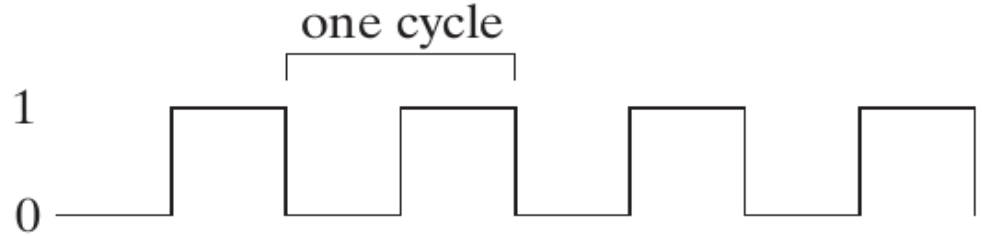
- **Clock synchronizes all internal operations of the CPU with other system components.**
- **The Control Unit(CU) coordinates the execution of instructions**
- **The Arithmetic Logic Unit (ALU) performs arithmetic and logical operations.**
- **The CPU is attached to the rest of the computer via pins.**
- **Most pins connect to the system bus**
- **A bus is a group of parallel wires for signals transmission.**
- **Four bus types: Address, Control, Data and I/O.**

2.1. GENERAL CONCCEPTS-4

2.1.2. BASIC MICROCOMPTER DESIGN

- **Clock pulses at a constant rate.**
- **Basic unit od machine instructions is a machine cycle or clock cycle**

Figure 2.1.2.2: Clock Cycle



- **A machine requires at least one clock cycle**
- **In 8088 processor, the multiply requires 50 clock cycles**
- **Instructions requiring memory access have wait states due speed differences.**

2.1. GENERAL CONCCEPTS-5

2.1.3. INSTRUCTION EXECUTION CYCLE

- **Executing a machine instruction requires three basic steps: *fetch, decode*, and *execute.***
- **FETCH:**
  - **The Control Unit fetches the next instruction from the queue.**
  - **It increases the Instruction Pointer (IP) aka Program Counter (PC).**
- **DECODE:**
  - **The Control Unit determines what the instruction will do.**
  - **Operands and signals on kind of operation are sent to the ALU.**
- ***FETCH OPERANDS*: if inputs in RAM; read from RAM into internal registers.**
- **EXECUTE:**
  - **ALU executes instruction using named registers and internal registers**
  - **ALU sends output to named registers and/or RAM**
  - **ALU updates status flags for processor state.**
- ***STORE OUTPUT:* The Control Unit uses write operation to store data to RAM or an I/O device.**

2.1. GENERAL CONCCEPTS-6

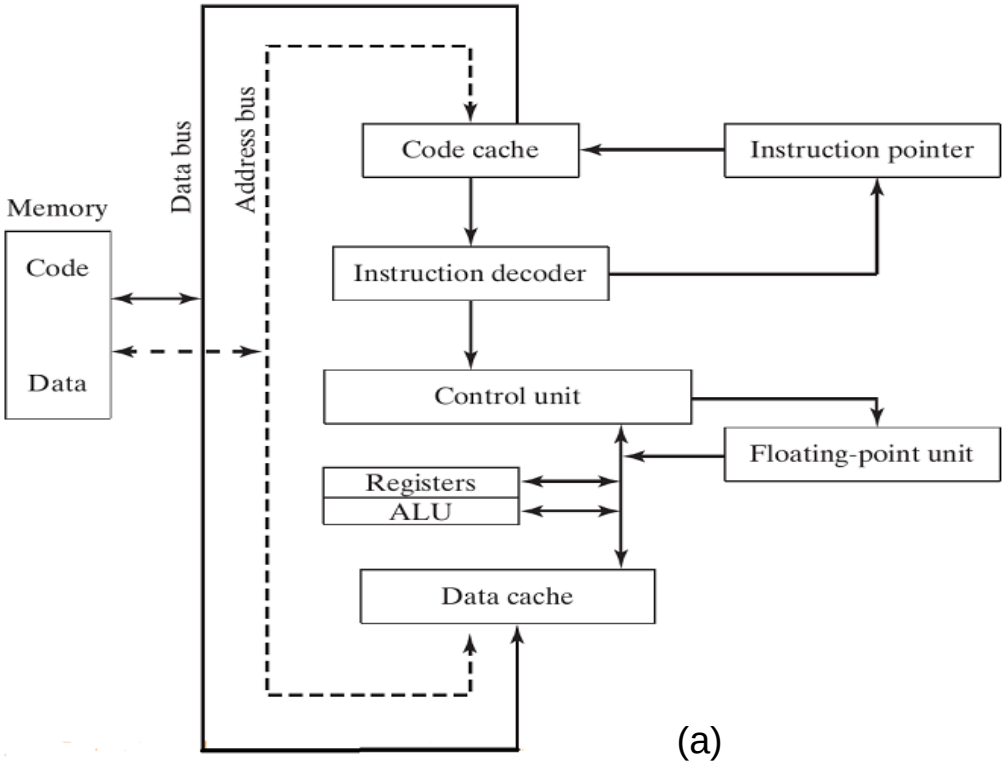
2.1.3. INSTRUCTION EXECUTION CYCLE – contd

- **The following pseudocode illustrates the above steps:**

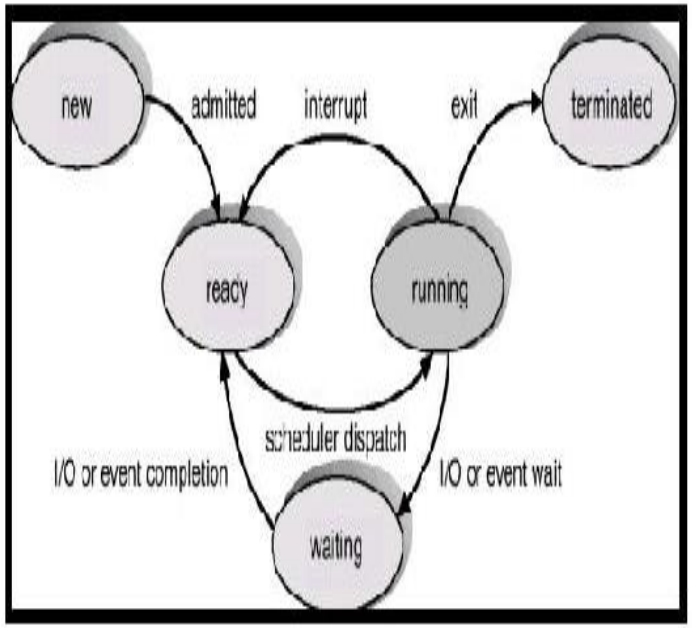
```
loop
    fetch next instruction
    advance the instruction pointer (IP)
    decode the instruction
    if memory operand needed, read value from memory
    execute the instruction
    if result is memory operand, write result to memory
continue loop
```

2.1. GENERAL CONCCEPTS-7

2.1.3. INSTRUCTION EXECUTION CYCLE – contd



(a)



(b)

Figure 2.1.3.1: (a) Simplified CPU Block Diagram
(b) Instruction Cycle State Diagram

2.1. GENERAL CONCCEPTS-8

2.1.4. READING FROM MEMORY

- **Program throughput depends on the speed of memory access.**
- **CPU waits for one or many clock cycles before the read operation completes; hence *wait states.***

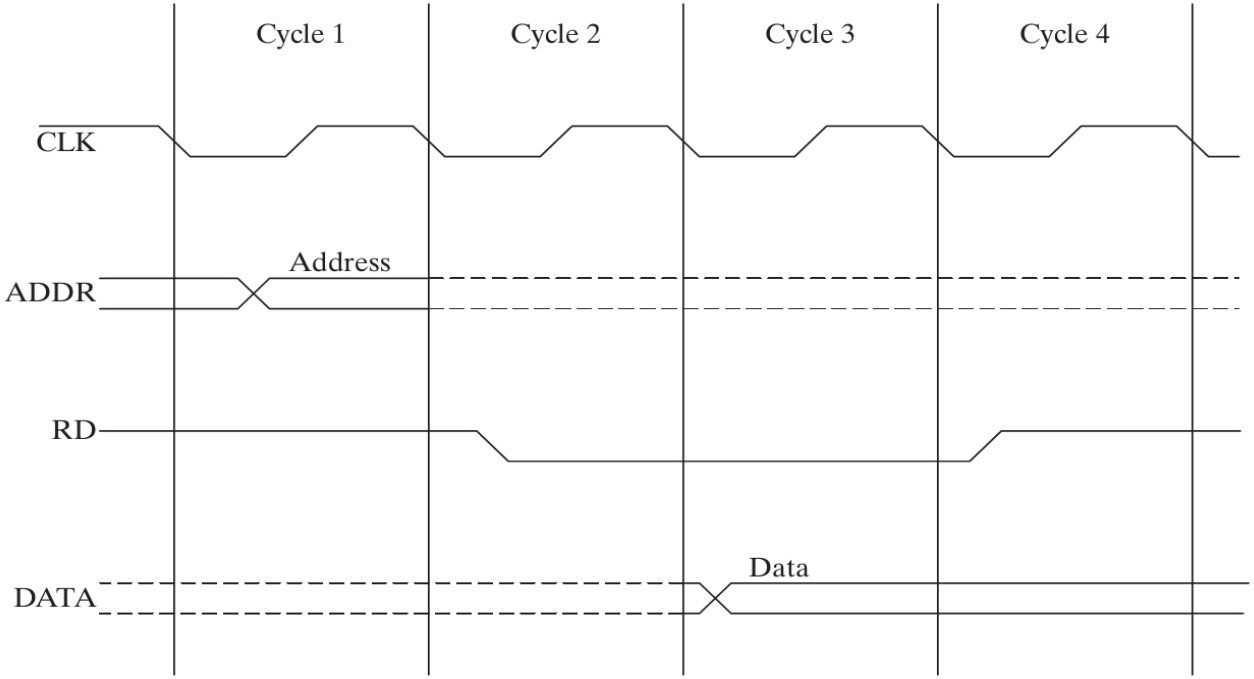


Figure 2.1.4.1: Memory Read Cycle

2.1. GENERAL CONCCEPTS-9

2.1.4. READING FROM MEMORY

- ***Cycle 1*: The address bits of the memory operand are placed on the address bus.**
- ***Cycle 2:* The line RD is low (0) to notify memory that data to be read.**
- ***Cycle 3*:The CPU waits one cycle, the memory controller places the operand on the data bus.**
- ***Cycle 4*:The read line goes to 1, signalling the CPU to read the data**

2.1. GENERAL CONCCEPTS-10

2.1.5. PROGRAM EXECUTION

- ***The OS tries to locate the program file in current disk directory.***
- ***If it cannot find the name there, it searches through paths***
- ***If the OS fails to find the program filename, error message.***
- ***If the program file is found, the OS retrieves all information about that program.***
- ***The OS determines the first instruction address and loads the program file into RAM***
- ***It allocates a block of memory to the program and enters a descriptor table.***

2.1. GENERAL CONCCEPTS-11

2.1.5. PROGRAM EXECUTION - contd

- *The OS begins execution of the program's first machine instruction.*
- *As soon as the program begins running, it is called a process.*
- *The OS assigns the process an identification number (PID)*
- *The process runs by itself.*
- *It is the OS's job to track the execution of the process*
- *When the process ends, it is removed from memory.*

2.1. GENERAL CONCCEPTS-12

2.1.6. MULTITASKING

- ***Most modern Operating Systems seemingly executes several tasks concurrently***
- ***The OS' Scheduler allocates CPU time slice (time slice) to each task.***
- ***By rapidly switching tasks, there is an illusion that the CPU is multitasking!***
- ***Only possible if a processor supports task switching***

2.1. GENERAL CONCCEPTS-13

2.1.6. MULTITASKING – contd

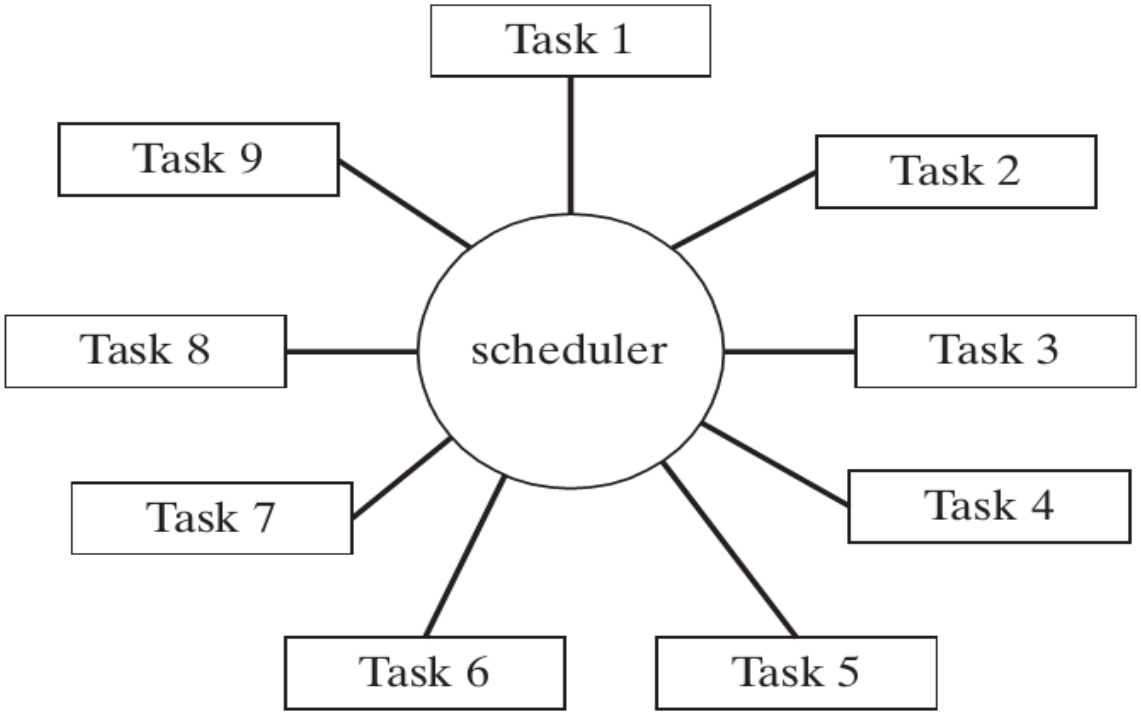


Figure 2.1.6.1: Round-Robin Scheduler

2.2. x86 ARCHITECTURE DETAILS-1

2.2.1. MODES OF OPERATION

- ***Protected Mode:*** **native state of the processor. Each program is allocated a segment.**
- ***Virtual-8086 Mode:*** **processor directly execute programs in Real-Address Mode in safe multitasking environment. e.g.: Windows XP executes multiple separate vitual-8086 sessions simultaneously.**
- ***Real-Address Mode:*** **offers programming environment with extra features such as switching to other modes. e.g.: blue screen crashes of Windows 98!**
- ***System Management Mode:*** **OS additionally manages power and security.**

2.2. x86 ARCHITECTURE DETAILS-2

2.2.2. BASIC EXECUTION ENVIRONMENT

- **ADDRESS SPACE*:***
  - **In 32-bit protected mode, a program points a linear address >=4GB**
  - **Starting with P6 processor, the technique called Extended Physical Addressing allows a total of 64GB of physical memory to be addressed**
  - **Read-address mode can only address 1MB**
  - **Processor in protected mode running multiple programs in virtual-8086 mode, each program has its own 1MB memory area.**
- **BASIC PROGRAM EXECUTION REGISTERS*:***
  - **There are 8 general purpose registers, 6 segment registers, 1 processor status flags register (EFLAGS), and an instruction pointer (EIP).**

2.2. x86 ARCHITECTURE DETAILS-3

2.2.2. BASIC EXECUTION ENVIRONMENT - contd

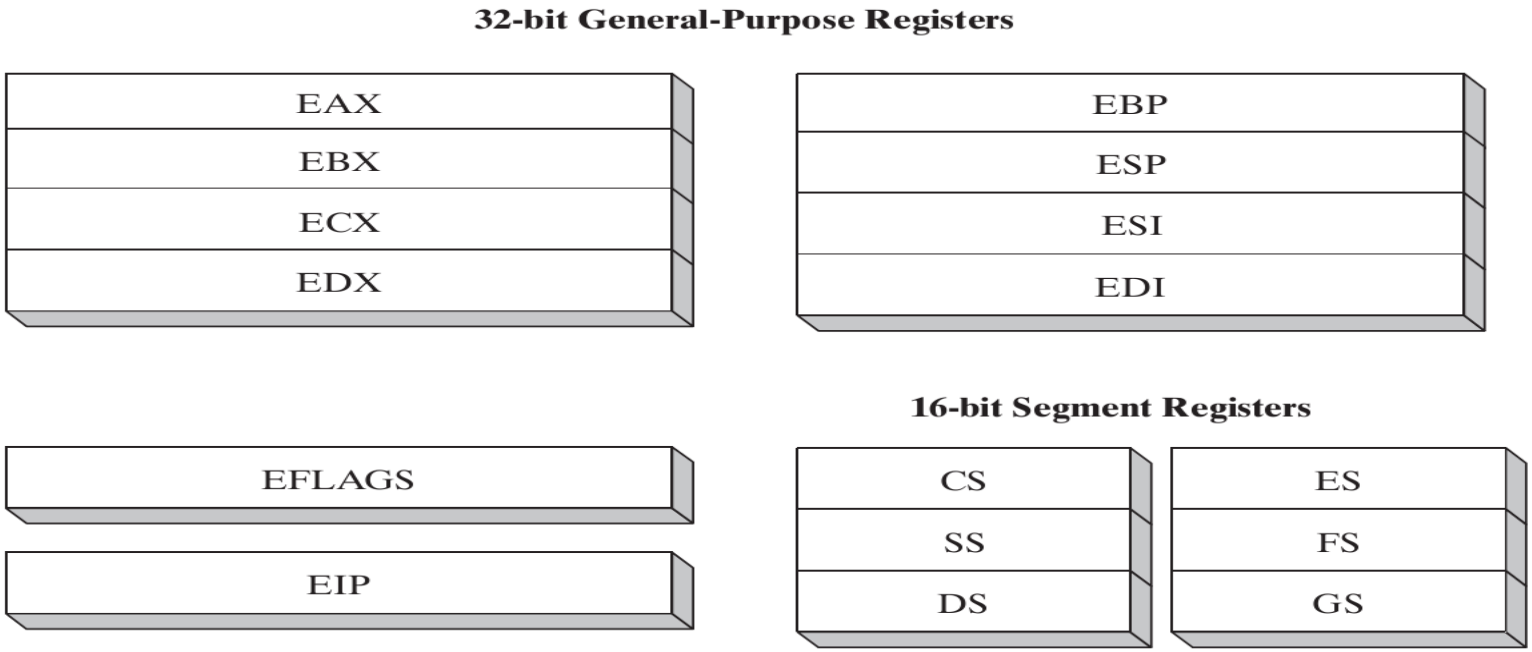


Figure 2.2.2.1: Basic Program Execution Registers

2.2. x86 ARCHITECTURE DETAILS-4

2.2.2. BASIC EXECUTION ENVIRONMENT - contd

- **GENERAL PURPOSE REGISTERS*:***

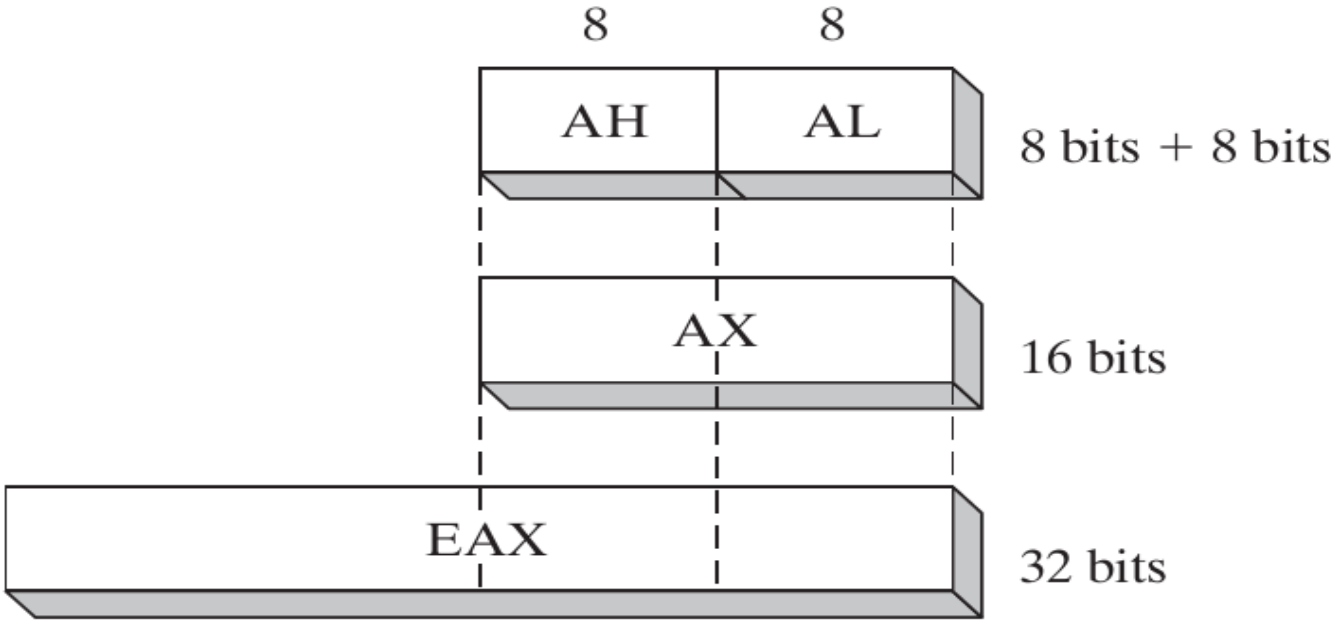


Figure 2.2.2.2: General Purpose Registers

2.2. x86 ARCHITECTURE DETAILS-5

2.2.2. BASIC EXECUTION ENVIRONMENT - contd

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

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

2.2. x86 ARCHITECTURE DETAILS-6

2.2.2. BASIC EXECUTION ENVIRONMENT - contd

- **SPECIALISED USES:**
  - **EAX (extended accumulator register) used by multiplication and division instructions**
  - **ECX used by the CPU as a loop counter**
  - **ESP (extended stack pointer) addresses data on the stack; rarely used for arithmetic or data transfer.**
  - **ESI (extended source index) and EDI (extended destination index), used by high-speed memory transfer instructions**
  - **EBP (extended frame pointer register), used by high-level language to reference function parameters and local variables on the stack**

2.2. x86 ARCHITECTURE DETAILS-7

2.2.2. BASIC EXECUTION ENVIRONMENT - contd

- **SEGMENT REGISTERS:**
  - **In real-address mode, 16-bit segment registers hold base addresses of segments**
  - **In protected mode, segment registers hold pointers to segment descriptor tables**
  - **Some segment registers hold program code**
  - **Other segment registers hold variables**
  - **Another segment register named stack segment holds local function variables and function parameters.**
- **INSTRUCTION POINTER (EIP):**
  - **It holds the address of the next instruction to be executed**
  - **Certain instructions manipulate EIP, causing the program to branch to a new location.**

2.2. x86 ARCHITECTURE DETAILS-8

2.2.2. BASIC EXECUTION ENVIRONMENT - contd

- **EFLAGS REGISTER:**
  - **Consists of bits that control the operation of the CPU or reflects the outcome of some CPU operations.**
  - **Some instructions test and manipulate individual processor flags**
  - **Flag is set = 1, flag is clear = 0**
- **CONTROL FLAGS:**
  - **Can cause the CPU to break execution**
  - **Interrupts when arithmetic overflows**
  - **Enter protected mode and virual-8086 mode**
  - **e.g.: Direction and Interrupt flags**
- **STATUS FLAGS:**
  - **Reflect outcomes of arithmetic or logical operations by the CPU**
  - **There are: Carry Flag (CF), Overflow Flag (OF), Sign Flag (SF), Zero Flag (ZF), Auxiliary Flag (AC) and Parity Flag (PF)**

2.2. x86 ARCHITECTURE DETAILS-9

2.2.2. BASIC EXECUTION ENVIRONMENT - contd

- **MMX REGISTERS:**
  - **MMX Technology added to Pentium processor to boost performance of multimedia and communications applications**
  - **The eight 64-bit MMX registers support SIMD (Single=Instruction, Multiple Data)**
  - **MMX Instructions operate in parallel on data in MMX registers**
  - **MMX registers names are aliases to registers used by the floating-point unit.**
- **XMM REGISTERS:**
  - **x86 architecture also eight 128-bit registers called XMM**
  - **Streaming of SIMD extensions to instruction set.**
  - **Certain instructions manipulate EIP, causing the program to branch to a new location.**

2.2. x86 ARCHITECTURE DETAILS-10

2.2.3. FLOATING-POINT UNIT (FPU)

- **Performs high-speed floating-point arithmetic**
- **From Intel486 onward, FPU is integrated into the CPU**
- **It has eight data registers: ST(0), ST(1), ST(2), … ST(7); control and pointer registers**

2.2. x86 ARCHITECTURE DETAILS-11

2.2.3. FLOATING-POINT UNIT (FPU) - contd

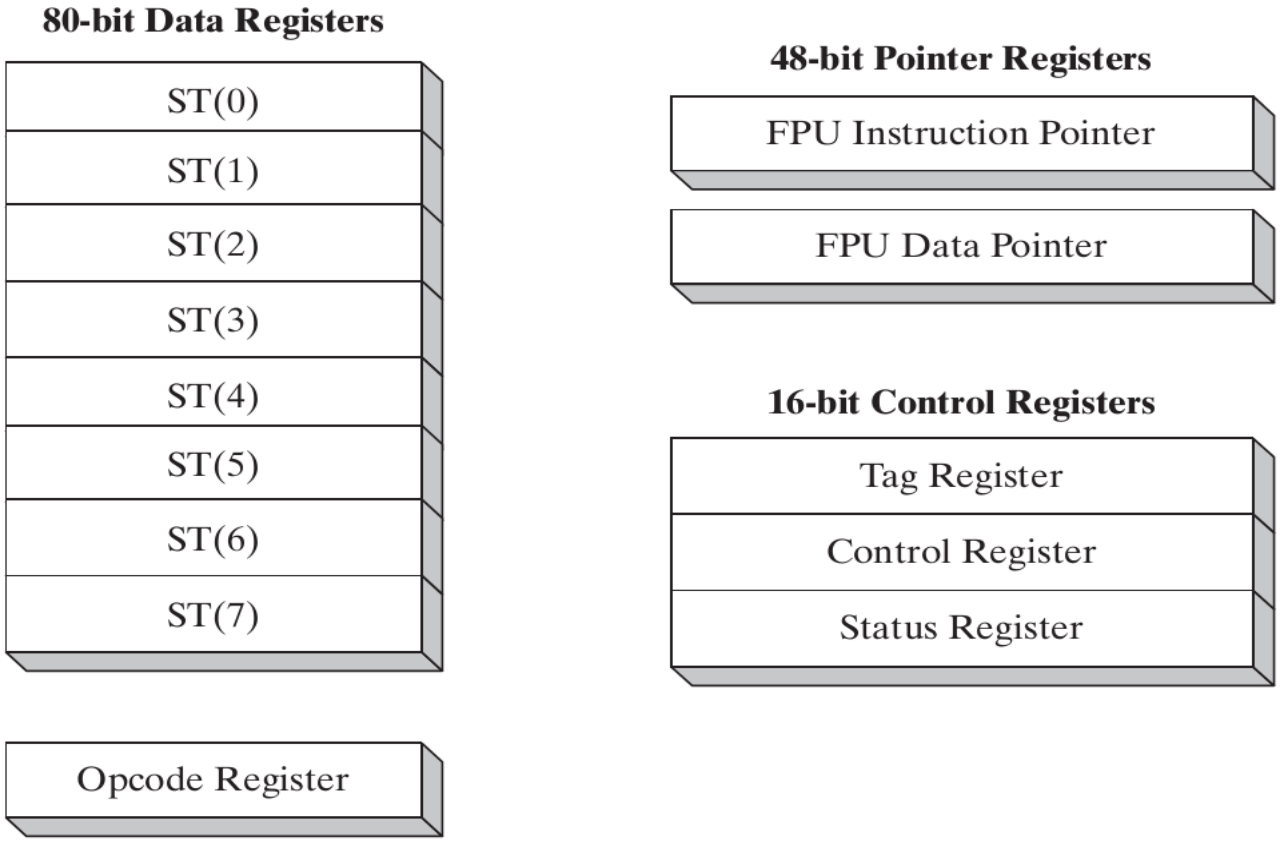


Figure 2.2.3.1: Floating-Point Unit Registers

2.2. x86 ARCHITECTURE DETAILS-12

2.2.4. OVERVIEW OF INTEL MICROPROCESSORS

- **IBM-PC HAD 16KB RAM with no hard disk**
- **Intel 8086 (1978) was the beginning of modern Intel architecture family:**
  - **16-bit registers**
  - **16-bit data bus**
  - **Segmented memory model =1MB**
- **IBM-PC (1980) had Intel 8088:**
  - **8-bit data bus**
  - **Less expensive to produce**
  - **Today, Intel 8088 is used in low-cost microcontrollers**
- **Backward Compatibility:**
  - **Since Intel 8086, older programs can run on newer hardware**
- **Intel 80286 Processor - first used in IBM-PC/AT:**
  - **Set new standards in processing and speed**
  - **First Intel processor to run in protected mode**
  - **16MB RAM using 24-bit address bus**

2.2. x86 ARCHITECTURE DETAILS-13

2.2.4. OVERVIEW OF INTEL MICROPROCESSORS, IA-32 PROCESSOR FAMILY (x86)

- **Intel 80386 Processor (1985) :**
  - **Introduced a 32-bit data registers and a 32-bit address bus and external data path**
  - **Internal data path moves data within the CPU**
  - **External data path (bus) moves data to and from memory and I/O devices**
  - **IA-32 processors address virtual memory larger than physical RAM**
  - **Each program is allocated 4GB linear address space**
- **Intel i486 Processor (1989):**
  - **Introduces a microarchitecture instruction set that pipelining**
- **Intel Pentium Processor (1993):**
  - **Introduces a superscalar design with two parallel execution pipelines**
  - **Uses a 32-bit address bus and a 64-bit internal data path**
  - **Introduces MMX technology**

2.2. x86 ARCHITECTURE DETAILS-14

2.2.4. OVERVIEW OF INTEL MICROPROCESSORS, INTEL64 FOR 64-BIT PROCESSING

- **Intel64 :**
  - **Implementation of x86-64 specification**
  - **Provides 64-bit linear address space**
  - **Individual processors generally implement leas than 64 bits**
  - **Physical address space can be greater than 64GB**
  - **Backward compatible**
  - **It has been used in Pentium Extreme, Intel Xeon, Celeron D, Pentium D, Core 2, Core i7, Atom processors and newer generations of Pentium 4**
  - **Added IA-32e mode for 64-bit processing**
- **IA-32e Mode:**
  - **Has two sub-modes**
  - **Compatibility Mode: legacy 16-bit and 32-bit applications run on 64-bit OS without recompilation. Operands are 16 and 32 bits and 4GB addressable RAM.**
  - **64-bit Mode: 64-bit addresses, 64-bit (and 32-bit) operands, a greater number of registers, and extensions to instruction set. Flat 64-bit linear address space.**
  - **Individual applications can run either in compatibility mode or 64-bit mode.**
  - **An application running in 64-bit mode cannot use segmented or read-address mode**

2.2. x86 ARCHITECTURE DETAILS-15

2.2.4. OVERVIEW OF INTEL MICROPROCESSORS, PROCESSOR FAMILIES

- **Intel Celeron—dual-core: 512KB L2 Cache, >= 2.2 GHz, 800 MHz bus**
- **Intel Pentium—dual-core: 2MB L2 Cache, 1.6 to 2.7 GHz, 800 MHz bus**
- **Core 2 Duo—2 processor cores, 1.8–3.33 GHz, 64 bit, 6 MB L2 cache**
- **Core 2 Quad—4 processor cores, up to 12 MB L2 cache, 1333 MHz front side bus**
- **Core i7—4 processor cores, >= 2.93 GHz, 8 processing threads, 8 MB smart cache, 3 channels DDR3 memory**

2.2. x86 ARCHITECTURE DETAILS-16

2.2.4. OVERVIEW OF INTEL MICROPROCESSORS, HYPERTHREADING AND MULTI-CORE PROCESSING

- **Hyper-Threading (HT) is a technology that divides a single physical processor into two logical processors.**
- **Dual-Core Processor: means a single IC that contains two complete physical processor chips in the same IC package**
- **Some Dual-Core processors incorporates HT technology**
- **Intel also offers multi-core packages**

2.2. x86 ARCHITECTURE DETAILS-17

2.2.4. OVERVIEW OF INTEL MICROPROCESSORS, CISC AND RISC

- **CISC: Complex Instruction Set Computer.**
- **Intel 8086 processor was the first to use CISC**
- **RISC: Reduced Instruction Set Computer, directly decodes and executes instructions using hardware.**

2.3. x86 MEMORY MANAGEMENT-1

2.3.1. REAL-ADDRESS MODE

- **A x86 Processor can access 1MB using 20-bit addresses.**
- **Memory scheme known as Segmented Memory**
- **Whole memory divide in segments of 64kB**
- **Segment values have always zero in last place**
- **There is always a 16-bit offset to every memory address**
- **E.g. The address 8000:0250 represents an offset of 250 inside the segment – linear address is 80250H**

2.3. x86 MEMORY MANAGEMENT-2

2.3.1. REAL-ADDRESS MODE - contd

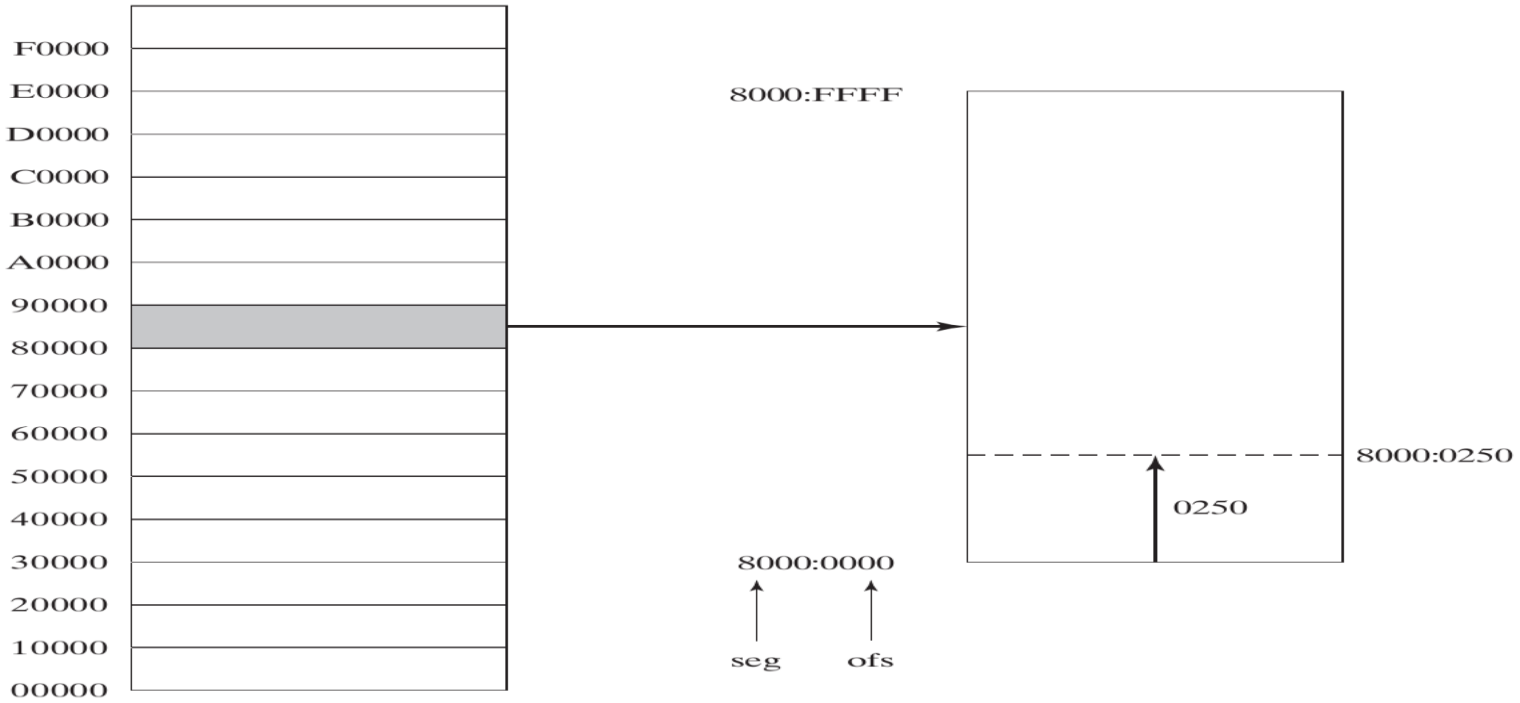


Figure 2.3.1.1: Segmented Memory Map, Real-Address Mode

2.3. x86 MEMORY MANAGEMENT-3

2.3.1. REAL-ADDRESS MODE - contd

- **Linear (absolute) address is 20 bits: 00000H - FFFFFH**
- **Programs cannot linear address directly**
- **Addresses are expressed using two 16-bit integers:**
  - **A 16-bit segment value, placed in one of the segment registers: CS, DS, ES, SS**
  - **A 16-bit offset value**

2.3. x86 MEMORY MANAGEMENT-4

2.3.1. REAL-ADDRESS MODE - contd

- **The CPU automatically converts a segment-offset address to a 20-bit linear address**
  - ***Example: segment-offset address is 08F1:0100***
    - **08F1h 10h = 08F10h (adjusted segment value)**
    - **Adjusted Segment value: 0 8 F 1 0**
    - **Add the offset: 0 1 0 0**
    - **Linear address: 0 9 0 1 0**

2.3. x86 MEMORY MANAGEMENT-5

2.3.1. REAL-ADDRESS MODE - contd

- **Typical program - three segments: code, data, and stack.**
- **Three segment registers, CS, DS, and SS**
  - **CS contains the 16-bit code segment address**
  - **DS contains the 16-bit data segment address**
  - **SS contains the 16-bit stack segment address**
  - **ES, FS, and GS can point to alternate data segments that supplement default data segment.**

2.3. x86 MEMORY MANAGEMENT-6

2.3.2. PROTECTED MODE

- **More powerful “native” processor mode.**
- **Linear Address Space – 4MB (00000000H – FFFFFFFFH)**
- **Microsoft Assembler, flat segmentation model**
- **Flat model: 32-bit integer for instruction or variable address**
- **CPU computes address translation behind scene**
- **Translation is apparent to application programmers**

2.3. x86 MEMORY MANAGEMENT-7

2.3.2. PROTECTED MODE - contd

- **Segment registers (CS, DS, SS, ES, FS, GS) point to segment descriptor tables**
- **Three segments: code, data, and stack (CS, DS and SS)**
  - **CS references the descriptor table for the code segment**
  - **DS references the descriptor table for the data segment**
  - **SS references the descriptor table for the stack segment**

2.3. x86 MEMORY MANAGEMENT-8

2.3.2. PROTECTED MODE - contd

- **Flat Segmentation Model**
  - **All segments mapped to the entire 32-bit physical address space of the computer.**
  - **Two segments are required: code and data.**
  - **Each segment is defined by a segment descriptor, a 64-bit integer stored in a table known as the Global Descriptor Table (GDT).**

2.3. x86 MEMORY MANAGEMENT-9

2.3.2. PROTECTED MODE - contd

- **Flat Segmentation Model - contd**
  - **Figure 2.3.2.1 below shows a segment descriptor whose base address field points to the location in memory (00000000).**
  - **In this figure, the segment limit is 0040.**
  - **The access field contains bits that determine how the segment can be used.**
  - **All modern operating systems based on x86 architecture use the flat segmentation model.**

2.3. x86 MEMORY MANAGEMENT-10

2.3.2. PROTECTED MODE - contd

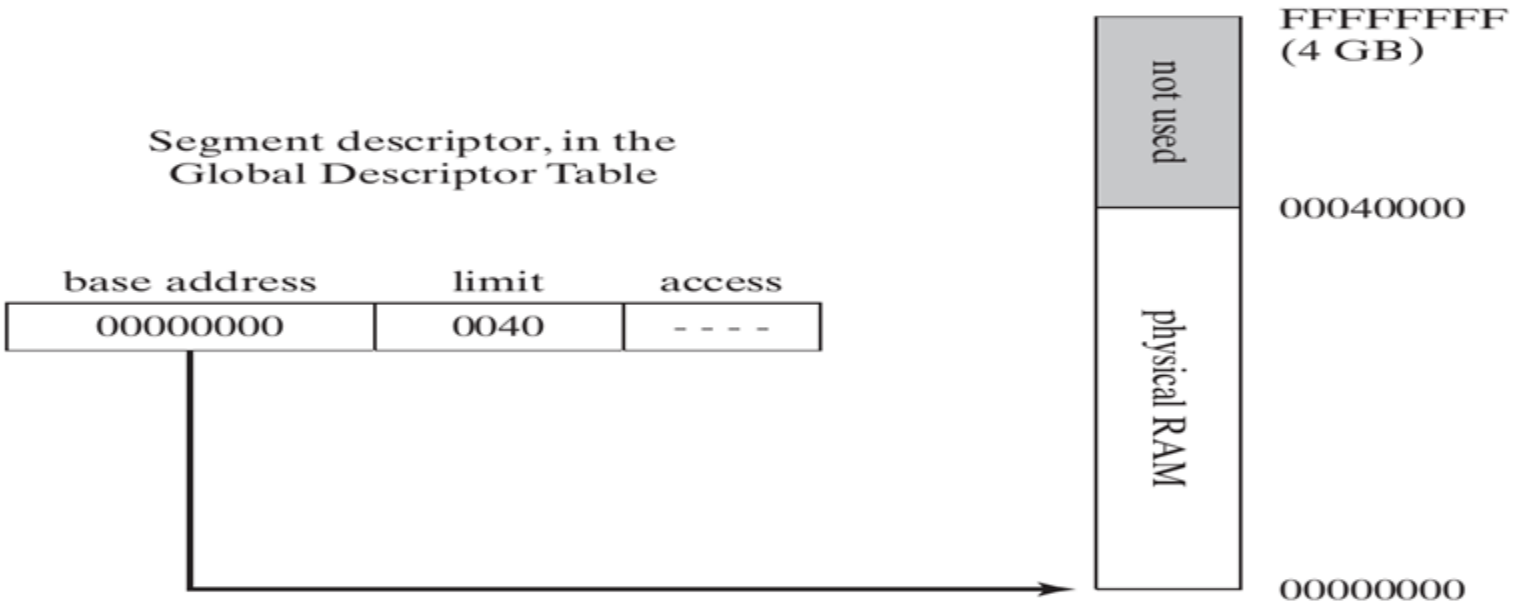


Figure 2.3.2.1: Flat Segmentation Model

2.3. x86 MEMORY MANAGEMENT-11

2.3.2. PROTECTED MODE - contd

- **Multi-Segment Model**
  - **Each program is given its own table of segment descriptors, called a Local Descriptor Table (LDT).**
  - **Each descriptor points to a segment, which can be distinct from all segments used by other processes.**
  - **Each segment has its own address space. In**

2.3. x86 MEMORY MANAGEMENT-12

2.3.2. PROTECTED MODE - contd

- **Multi-Segment Model - contd**
  - **In Figure 2.3.2.2, each entry in the LDT points to a different segment in memory.**
  - **Each segment descriptor specifies the exact size of its segment.**
  - **For example, the segment beginning at 3000 has size 2000H, which is computed as (0002 x1000H).**
  - **The segment beginning at 8000H has size A000H.**

2.3. x86 MEMORY MANAGEMENT-13

2.3.2. PROTECTED MODE - contd

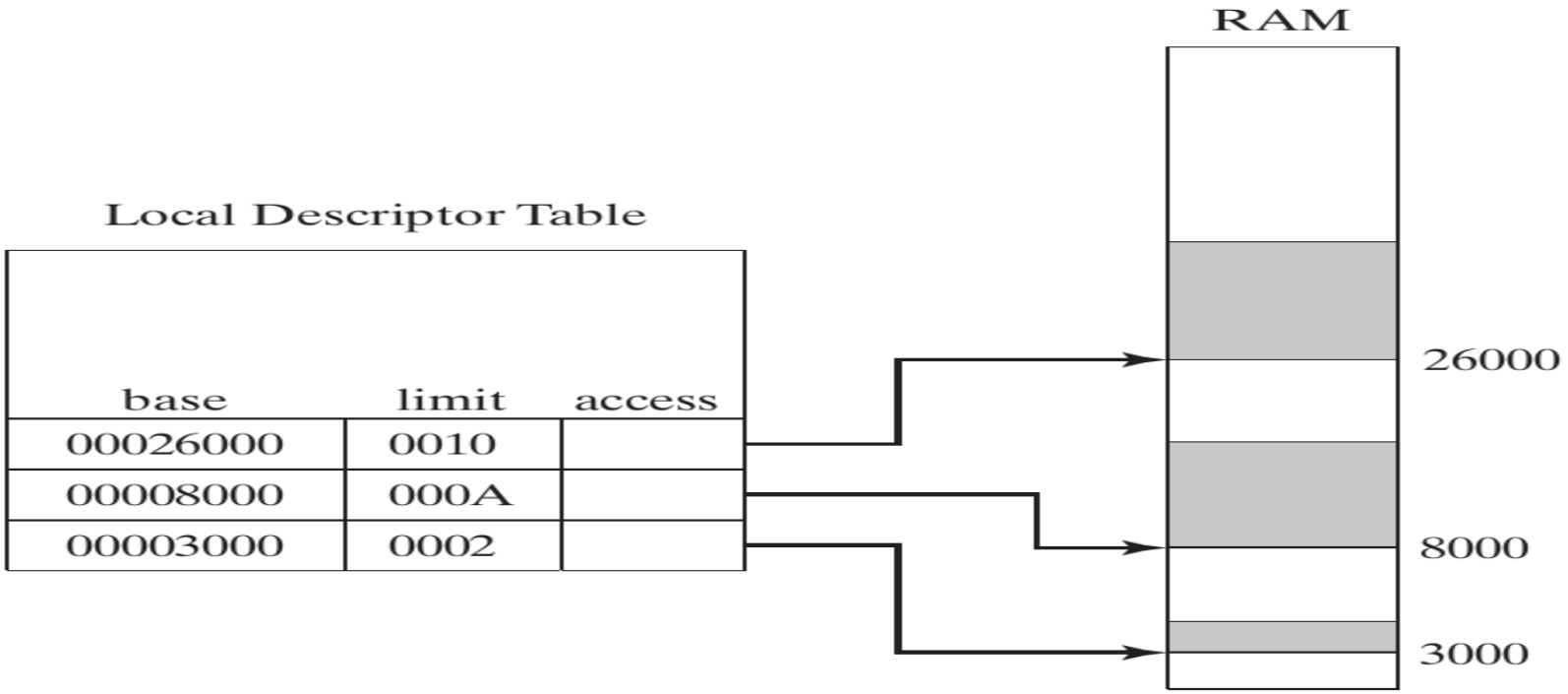


Figure 2.3.2.2: Multi-Segment Model

2.3. x86 MEMORY MANAGEMENT-14

2.3.2. PROTECTED MODE - contd

- **Paging**
  - **A feature that permits segments to be divided into 4,096-byte blocks of memory called pages.**
  - **Paging permits the total memory used by all concurrent programs to be much larger than the computer's physical memory.**
  - **Complete collection of pages mapped by the operating system is called virtual memory.**

2.3. x86 MEMORY MANAGEMENT-15

2.3.2. PROTECTED MODE - contd

- **Paging - contd**
  - **Operating systems have utility programs named virtual memory managers.**
  - **Disk storage is cheap and plentiful.**
  - **Paging provides the illusion that memory is almost unlimited in size.**
  - **A program which relies on paging, is slower.**
  - **When a task is running, parts of it can be stored on disk if they are not currently in use.**

2.3. x86 MEMORY MANAGEMENT-16

2.3.2. PROTECTED MODE - contd

- **Paging - contd**
  - **Parts of the task are paged (swapped) to disk.**
  - **Other actively executing pages remain in memory.**
  - **The processor issues a page fault, causing the page or pages containing the required code or data to be loaded back into memory.**
  - **To see how this works, find a computer with somewhat limited memory and run many large applications at the same time.**

2.4 COMPONENTS OF A TYPICAL x86 COMPUTER - 1

2.4.1. MOTHERBOARD

- **The heart of a microcomputer is its motherboard, a flat circuit board**
- **The various components are connected to each other by a bus**
- **The following components have traditionally been found on PC motherboards:**
  - **A CPU socket.**
  - **Memory slots (SIMM or DIMM)**
  - **BIOS (basic input-output system) computer chips**
  - **CMOS RAM, with a small circular battery to keep it powered**
  - **Connectors for mass-storage devices such as hard drives and CD-ROMs**
  - **USB connectors for external devices**
  - **Keyboard and mouse ports**
  - **PCI bus connectors for sound cards, graphics cards, data acquisition boards, and other input-output devices**

2.4 COMPONENTS OF A TYPICAL x86 COMPUTER - 2

2.4.1. MOTHERBOARD - contd

- **The following components are optional:**
  - **Integrated sound processor**
  - **Parallel and serial device connectors**
  - **Integrated network adapter**
  - **AGP bus connector for a high-speed video card**

2.4 COMPONENTS OF A TYPICAL x86 COMPUTER - 3

2.4.1. MOTHERBOARD - contd

- **Following are some important support processors in a typical system:**
  - **The Floating-Point Unit (FPU) The 8284/82C284 Clock Generator.**
  - **The clock generator synchronizes the CPU and the rest of the computer.**
  - **The 8259A Programmable Interrupt Controller(PIC)**
  - **These devices interrupt the CPU and make it process their requests immediately.**
  - **The 8253 Programmable Interval Timer/Counter interrupts the system 18.2 times per second.**
  - **It is also responsible for constantly refreshing memory The 8255 Programmable Parallel Port.**
  - **This port is commonly used for printers**

2.4 COMPONENTS OF A TYPICAL x86 COMPUTER - 4

2.4.1. MOTHERBOARD - contd

- **PCI and PCI Express Bus Architectures**
  - **The PCI (Peripheral Component Interconnect) bus provides a bridge between the CPU and other system devices such as hard drives, memory, video controllers, sound cards, and network controllers.**
  - **More recently, the PCI Express bus provides two-way serial connections between devices, memory, and the processor.**
  - **It carries data in packets, similar to networks, in separate "lanes."**
  - **It is widely supported by graphics controllers, and can transfer data at about 4 GB/s.**

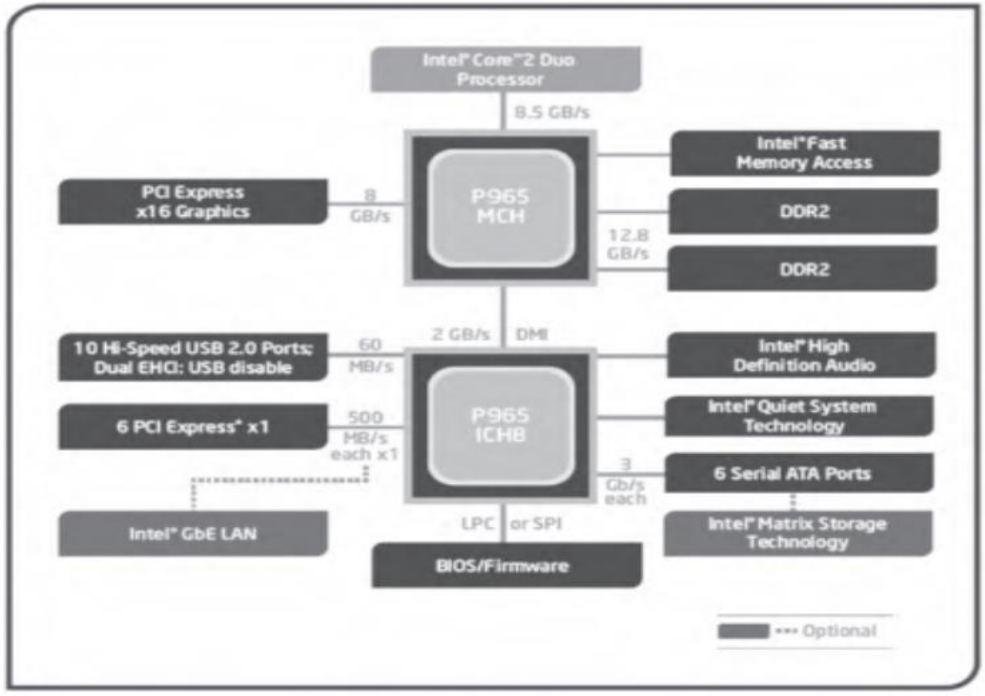
2.4 COMPONENTS OF A TYPICAL x86 COMPUTER - 5

2.4.1. MOTHERBOARD - contd

- **Motherboard Chipset**
  - **A collection of processor chips designed to work together on a specific type of motherboard. capabilities, or reduce power consumption.**
  - **The Intel P965 Express Chipset can be used as an example.**
  - **It is used in desktop PCs, with either an Intel Core 2 Duo or Pentium D processor.**
  - **Here are some of its features:**
    - **Intel Fast Memory Access uses an updated Memory Controller Hub (MCH).**
    - **It can access dual-channel DDR2 memory, at an 800 MHz clock speed.**
    - **An I/O Controller Hub (Intel ICH8/R/DH) uses Intel Matrix Storage Technology (MST) to support six Serial ATA devices (disk drives).**
    - **Support for 10 USB ports, six PCI express slots, networking, Intel Quiet System technology.**
    - **A high definition audio chip provides digital sound capabilities.**

2.4 COMPONENTS OF A TYPICAL x86 COMPUTER - 6

2.4.1. MOTHERBOARD – contd



Source: The Intel P965 Express Chipset (product brief),
© 2006 by Intel Corporation, used by permission.

Figure 2.4.1.1: Intel 965 Express Chipset Block Diagram

2.4 COMPONENTS OF A TYPICAL x86 COMPUTER - 7

2.4.2. VIDEO OUTPUT

- **The video adapter controls the display of text and graphics.**
- **It has two components: the video controller and video display memory.**
- **The Video controller writes data to the video display RAM before sending it to the monitor.**
- **The video controller is itself a special-purpose microprocessor, relieving the primary CPU of the job of controlling video hardware.**
- **Older Cathode-ray tube (CRT) video displays used a technique called raster scanning to dis-play images.**
- **A direct digital Liquid Crystal Display (LCD) panel, considered standard today, receives a digital bit stream directly from the video.**
- **Digital displays generally display sharper**

2.4 COMPONENTS OF A TYPICAL x86 COMPUTER - 8

2.4.3. MEMORY

- **Different types are:**
  - **ROM is permanently burned into a chip and cannot be erased.**
  - **EPROM can be erased slowly with ultraviolet light and reprogrammed.**
  - **DRAM, commonly known as main memory,**
  - **Some systems use ECC (error checking and correcting) memory.**
  - **SRAM is used primarily for expensive, high-speed cache memory. It does not have to be refreshed.**
  - **CPU cache memory is comprised of SRAM.**
  - **VRAM holds video data. It is dual ported**
  - **CMOS RAM on the system motherboard stores system setup information.**

2.4 COMPONENTS OF A TYPICAL x86 COMPUTER - 9

2.4.4. I/O PORTS AND DEVICE INTERFACE

- **Universal Serial Bus (USB)**
- **Parallel Port (LPT1, LPT2, LPT3)**
- **ATA Host Adapters**
- **SATA Host Adapters**
- **FireWire (800MB/s)**
- **Bluetooth**
- **Wi-Fi aka wireless Ethernet**

2.5 INPUT-OUTPUT SYSTEM - 1

2.5.1. LEVELS OF I/O ACCESS

- **High-level language functions**
- **Operating system**
- **BIOS**
- **Device Drivers Device drivers**

2.5 INPUT-OUTPUT SYSTEM - 2

2.5.1. LEVELS OF I/O ACCESS - contd

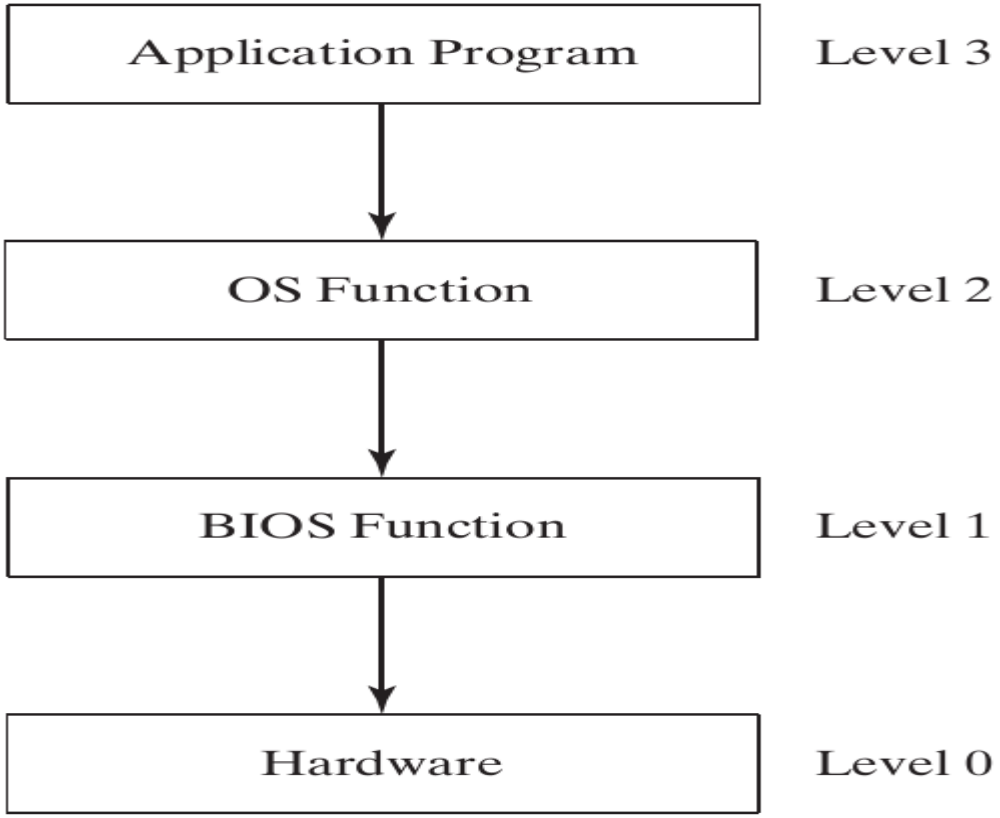


Figure 2.5.1.1: Access Levels for Input-Output Operations