

UCEC1044 Basic Microprocessor

# Architecture of Microprocessor

Y.C.See

chark97@hotmail.com

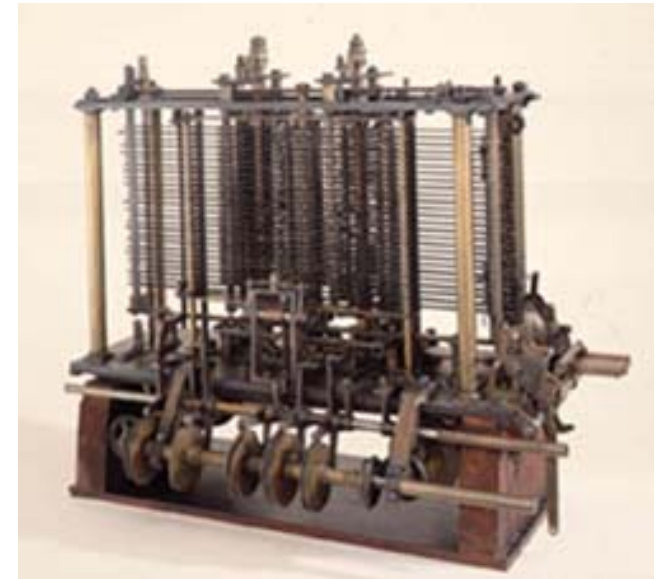


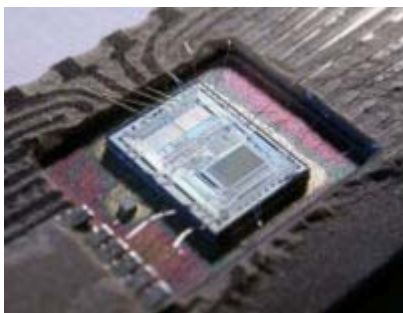
# Course Introduction

- Timetable
- Contents
- Schedule
- Assessments
- Laboratories
- Others

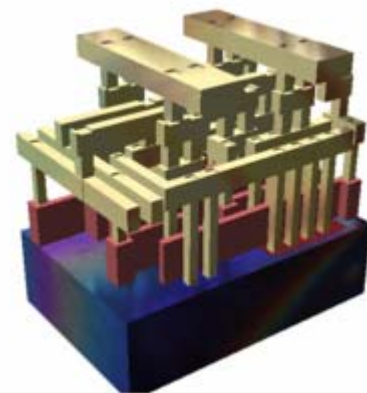
# Mechanical Age

- Charles Babbage was the pioneer of mechanical computing machinery
  - Analytical Engine in 1832
    - Assisted by Augusta Ada King
    - Steam powered, 50 000 components
    - Input via punch cards, control unit, memory unit
    - to calculate a series of numerical values and automatically print the results.





# Electrical Age



- There are three widely recognized generations of electrical-based, digital computers:

- ☐ First:

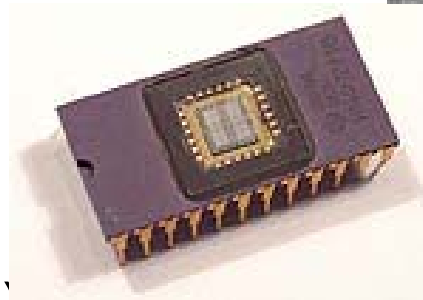
- Vacuum tubes

- ☐ Second:

- Transistors

- ☐ Third:

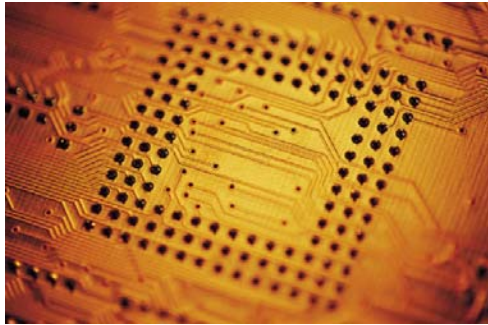
- Integrated circuits (ICs)





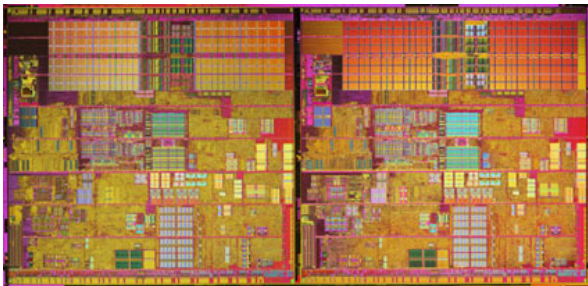
# Modern Computers

- Supercomputers
- Mainframes
- Minicomputers (workstations, servers)
- Microcomputers (PCs)
- Microcontrollers (Intel MCS-51 families, Zilog, HCS-12 Motorola etc.)



# Microprocessor

1971 to 2007



230 million transistors, 206  
mm<sup>2</sup> die size, 90 nm  
production process



Source: <http://www.intel.com/intel/intelis/museum/>

Family	Trade Name (Code Name for Future Chips)	Clock Frequency in MegaHertz**	Millions of Instructions per Second	Date of Introduction	Number of Transistors	Design Rule (Pixel Size)	Address Bus Bits
80986	Projected Roadmap	24,000.0 MHz	+125,000. MIPS	2007	1 billion	0.045 micron	64 bit
80886	(Northwood)	3,000.0 MHz	TBA	2003	TBA	0.13 micron	64 bit
80886	(Madison)	TBA	TBA	2003	TBA	0.13 micron	64 bit
80886	(Deerfield)***	TBA	TBA	2002Q2	TBA	0.13 micron	64 bit
80886	(McKinley)	1,000.0 MHz	TBA	2002Q1	TBA	0.18 micron	64 bit
80786	Itanium (Merced)	800.0 MHz	+2,500.00 MIPS	May 29, 2001	30 / 300 M	0.18 micron	64 bit
80686	Pentium 4	1,500.0 MHz	*1,500.00 MIPS	November 20, 2000	42 million	0.18 micron	32 bit
80686	Pentium III	1,000.0 MHz	*1,000.00 MIPS	March 1, 2000	28.1 million	0.18 micron	32 bit
80686	P III Xeon	733.0 MHz	*733.00 MIPS	October 25, 1999	28.1 million	0.18 micron	32 bit
80686	Mobile P II	400.0 MHz	*400.00 MIPS	June 14, 1999	27.4 million	0.18 micron	32 bit
80686	P III Xeon	550.0 MHz	*550.00 MIPS	March 17, 1999	9.5 million	0.25 micron	32 bit
80686	Pentium III	500.0 MHz	*500.00 MIPS	February 26, 1999	9.5 million	0.25 micron	32 bit
80686	P II Xeon	400.0 MHz	*400.00 MIPS	June 29, 1998	7.5 million	0.25 micron	32 bit
80686	Pentium II	333.0 MHz	*333.00 MIPS	January 26, 1998	7.5 million	0.25 micron	32 bit
80686	Pentium II	300.0 MHz	*300.00 MIPS	May 7, 1997	7.5 million	0.35 micron	32 bit
80586	Pentium Pro	200.0 MHz	*200.00 MIPS	November 1, 1995	5.5 million	0.35 micron	32 bit
90586	Pentium	133.0 MHz	*133.00 MIPS	June 1995	3.3 million	0.35 micron	32 bit
80586	Pentium	90.0 MHz	*90.00 MIPS	March 7, 1994	3.2 million	0.60 micron	32 bit
80586	Pentium	60.0 MHz	*60.00 MIPS	March 22, 1993	3.1 million	0.80 micron	32 bit
80486	80486 DX2	50.0 MHz	*50.00 MIPS	March 3, 1992	1.2 million	0.80 micron	32 bit
80486	486 DX	25.0 MHz	20.00 MIPS	April 10, 1989	1.2 million	1.00 micron	32 bit
80386	386 DX	16.0 MHz	5.00 MIPS	October 17, 1985	275,000	1.50 micron	16 bit
80286	80286	6.0 MHz	0.90 MIPS	February 1982	134,000	1.50 micron	16 bit
8086	8086	5.0 MHz	0.33 MIPS	June 8, 1978	29,000	3.00 micron	16 bit
8080	8080	2.0 MHz	0.64 MIPS	April 1974	6,000	6.00 micron	8 bit
8008	8008	.2 MHz	0.06 MIPS	April 1972	3,500	10.00 micron	8 bit
4004	4004	.1 MHz	0.06 MIPS	November 15, 1971	2,300	10.00 micron	4 bit

\* Approximately one instruction per processor clock cycle // + /starting with Itanium, the chips have multiple floating point processors per chip

\*\* 1 KHz (KiloHertz) = 1 thousand cycles per second; 1 MegaHertz = 1 thousand KiloHertz; 100 KHz = .1 MHz,

1 GHz (GigaHertz) = 1 billion cycles per second; 1 GigaHertz = 1 thousand MegaHertz

TBA To be announced, Pentium 4 was formerly code named Willamette \*\*\* Deerfield is a low cost version of Madison.

<http://www.esc-ca.com/processors/intel/future.htm> (one source of data for future microprocessors)

<http://www.Intel.com/pressroom/kits/processors/quickref.htm> (source of data for released microprocessors)



## Extreme

**Intel® Pentium® processor Extreme Edition 965A**

Dual-core, 2x2 MB L2 cache, 3.73 GHz, 1066 MHz FSB  
Intel® 975X Express Chipset

**Intel® Core™2 Extreme processor**

Next generation Extreme Edition processor  
Dual-core  
Intel® 975X Express Chipset

## Dual-Core Processors

**Intel® Core™2 Duo processor**

Next generation dual-core processor  
Intel® 975X Express Chipset  
Intel® 965 Express Chipset family

**Intel® Pentium® D processor 960A**

Dual-core, 2x2 MB L2 cache, 3.60 GHz, 800 MHz FSB  
Intel® 975X Express Chipset  
Intel® 955X Express Chipset  
Intel® 945 Express Chipset family

**Intel® Pentium® D processor 960A or greater**

Next generation dual-core processor  
Intel® 975X Express Chipset  
Intel® 955X Express Chipset  
Intel® 965 Express Chipset family

## Single-Core Processors

**Intel® Pentium® 4 processor 670A supporting Hyper-Threading Technology†**

2 MB L2 cache, 3.80 GHz, 800 MHz FSB  
Intel® 975X Express Chipset  
Intel® 955X Express Chipset  
Intel® 945 Express Chipset family

**Intel® Pentium® 4 processor 541A supporting Hyper-Threading Technology† or greater**

1 MB L2 cache, 3.20 GHz, 800 MHz FSB  
Intel® 975X Express Chipset  
Intel® 955X Express Chipset  
Intel® 945 Express Chipset family

**Intel® Celeron® D processor 356A**

512 KB L2 cache, 3.33 GHz, 533 MHz FSB  
Intel® 945 Express Chipset family  
Intel® 915 Express Chipset family

**Intel® Celeron® D processor 360A or greater**

512 KB L2 cache, 3.46 GHz, 533 MHz FSB  
Intel® 945 Express Chipset family  
Intel® 915 Express Chipset family

# INTEL ROAD MAP

[www.intel.com](http://www.intel.com)



NEAR-TERM PRODUCT OUTLOOK | NOVEMBER 2005 → NOVEMBER 2006

### SERVERS & WORKSTATIONS



### MOBILE



### DESKTOP



### EMBEDDED



# AMD Roadmap

[www.amd.com](http://www.amd.com)

# Intel perspective: Silicon is the *Engine*

Platforms

Physical  
Science

Meets

Converged  
Computing and  
Communications

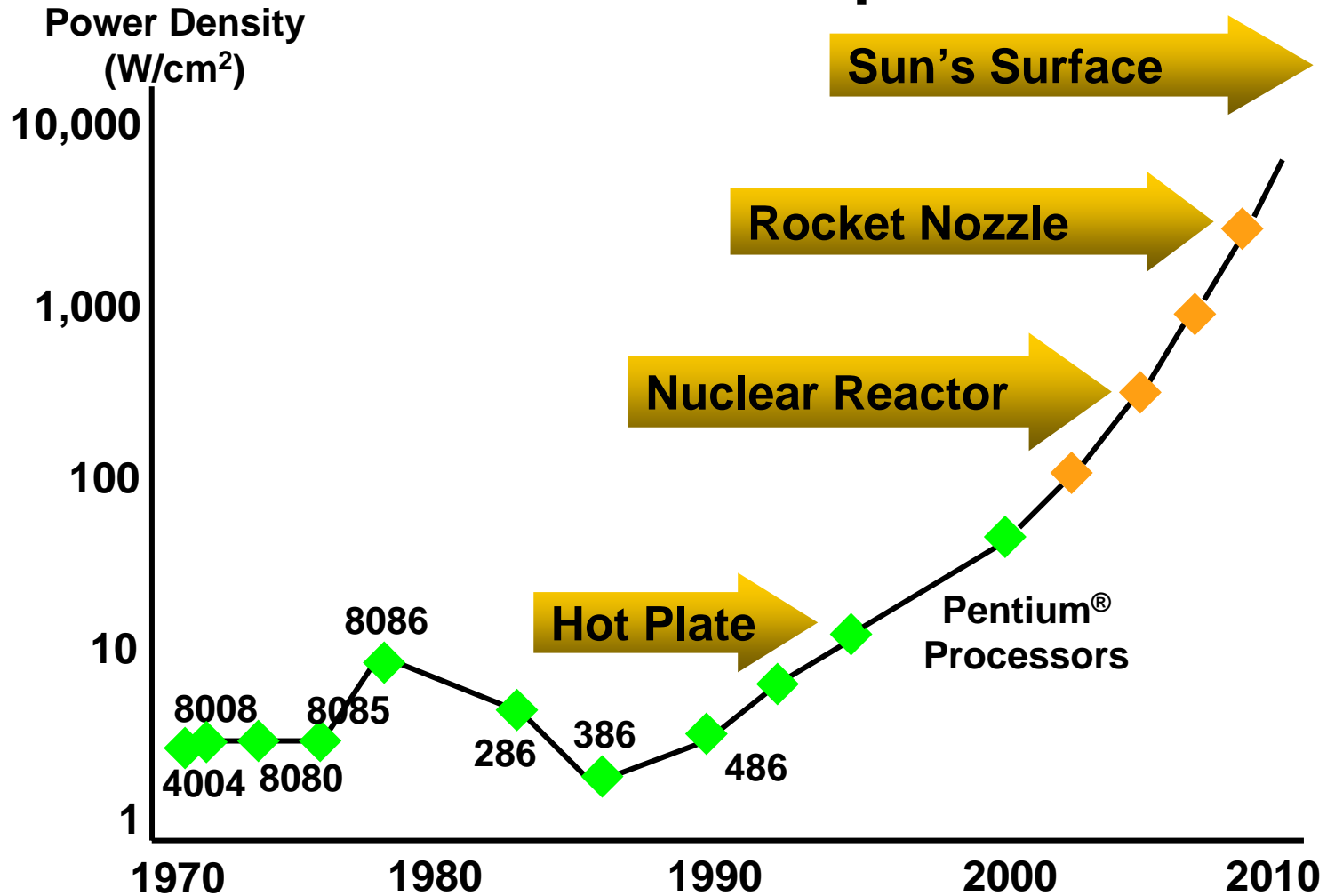
Microprocessors

Information  
Science

Memory

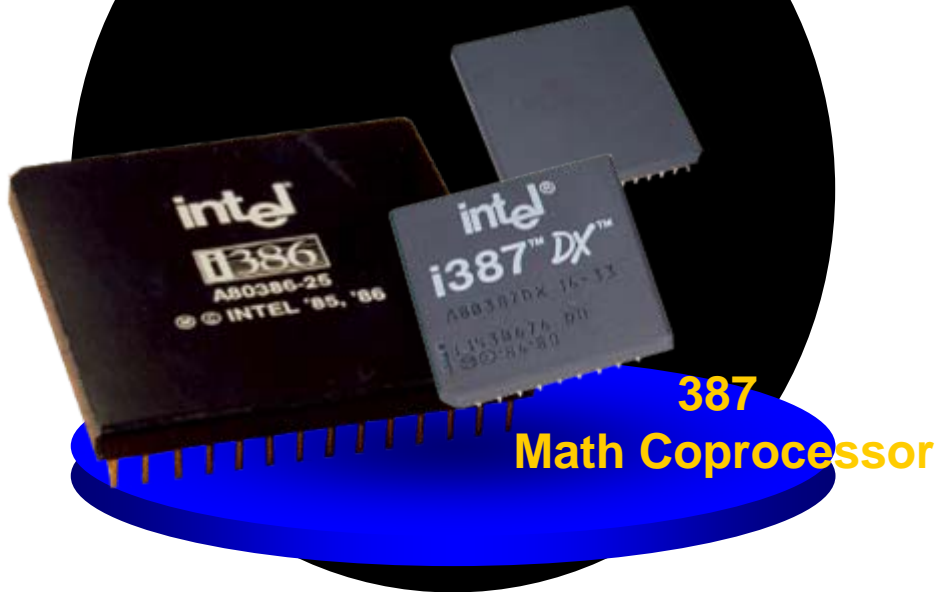
## Entering Intel's Third Era

# Power consumption

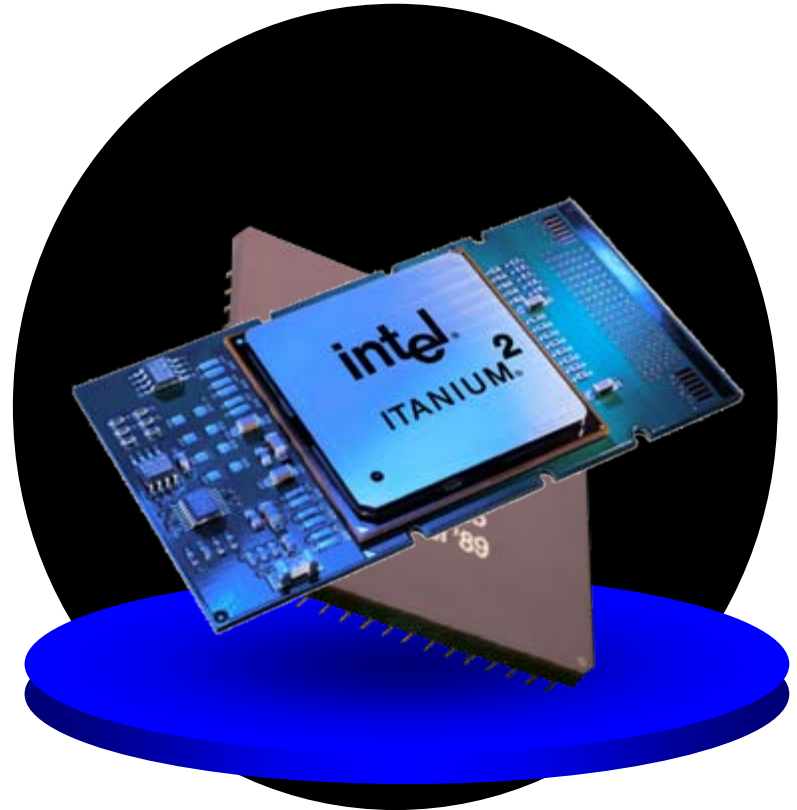


# Integration example: Processors

Off Chip Cache:  
82395 DX



1985  
Intel 386™ Processor



1989  
Intel Pentium™ Processors

# Integration example: Networking



1994

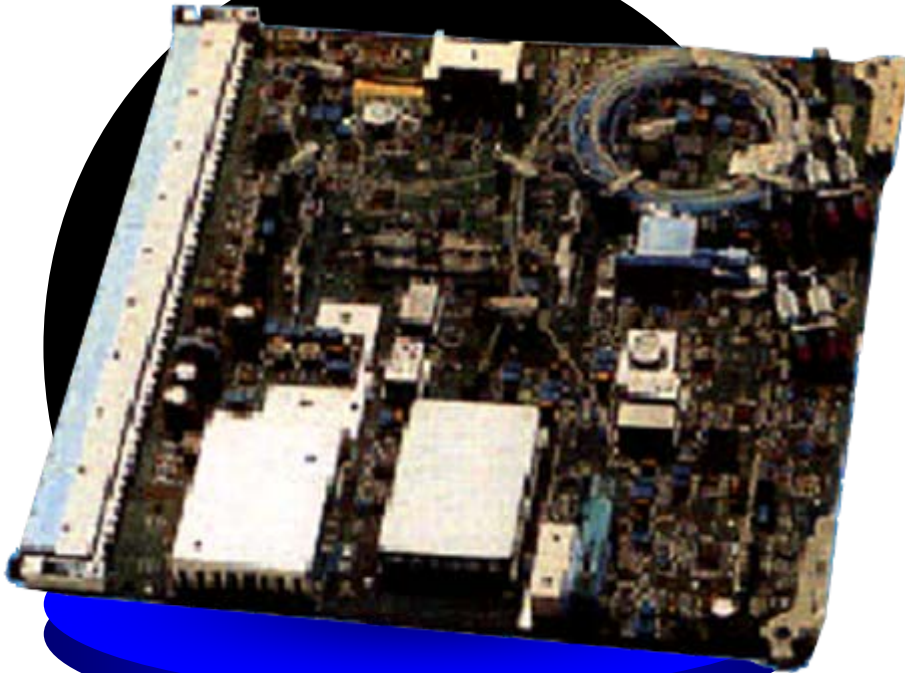
Fast Ethernet Card: 10 chips



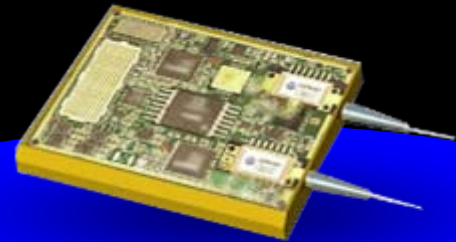
1997

1 chip

# Integration example: Optical Networking

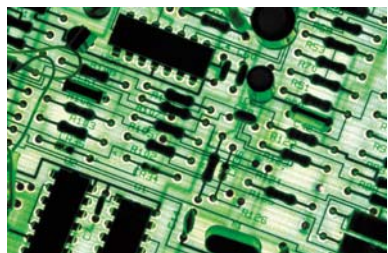


**2001**  
**100's of chips**



**2003**  
**10+ Chips**





# System Bus (1)



- Collection of wires on which electrical signals pass between components in the system.
- 3 major busses: the *address* bus, the *data* bus, and the *control* bus.
- These busses vary from processor to processor.





# Processor Specifications (1)



## ■ Data I/O Bus

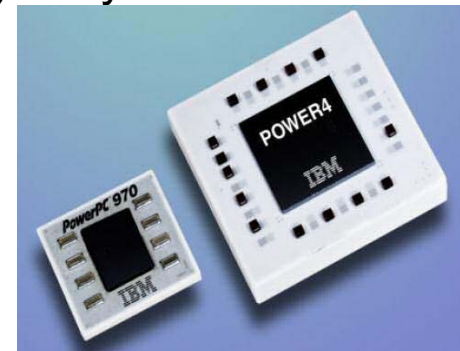
- data that can carry in a time –8, 16, 32 or 64 bits, larger enables greater throughput

## ■ Address bus

- memory location to which the data being read or write. More wires (digits), greater the maximum amount of RAM a chip can address.

## ■ Control bus

- eclectic collection of signals that control how the processor communicates with the rest of the system
- Carries timing signals (and more) to synchronize CPU to external circuitry
- R/W



# Processor Specifications (2)

Processor Family	Address Bus	Bytes	Kilobytes (KB)	Megabytes (MB)	Giga-bytes (GB)	Tera-bytes (TB)
8088 8086	20-bit	1,048,576	1,024	1	—	—
286 386SX	24-bit	16,777,216	16,384	16	—	—
386DX 486 586	32-bit	4,294,967,296	4,194,304	4,096	4	—
686 786	36-bit	68,719,476,736	67,108,864	65,536	64	—
Itanium	44-bit	17,592,186,044,416	17,179,869,184	16,777,216	16,384	16



# Architectures -I

- **Harvard architecture** - separate data and instruction busses, allowing transfers to be performed simultaneously on both busses.
- **Von Neumann architecture** - only one bus which is used for both data transfers and instruction fetches, and therefore data transfers and instruction fetches must be scheduled - they can not be performed at the same time

# Architectures -II

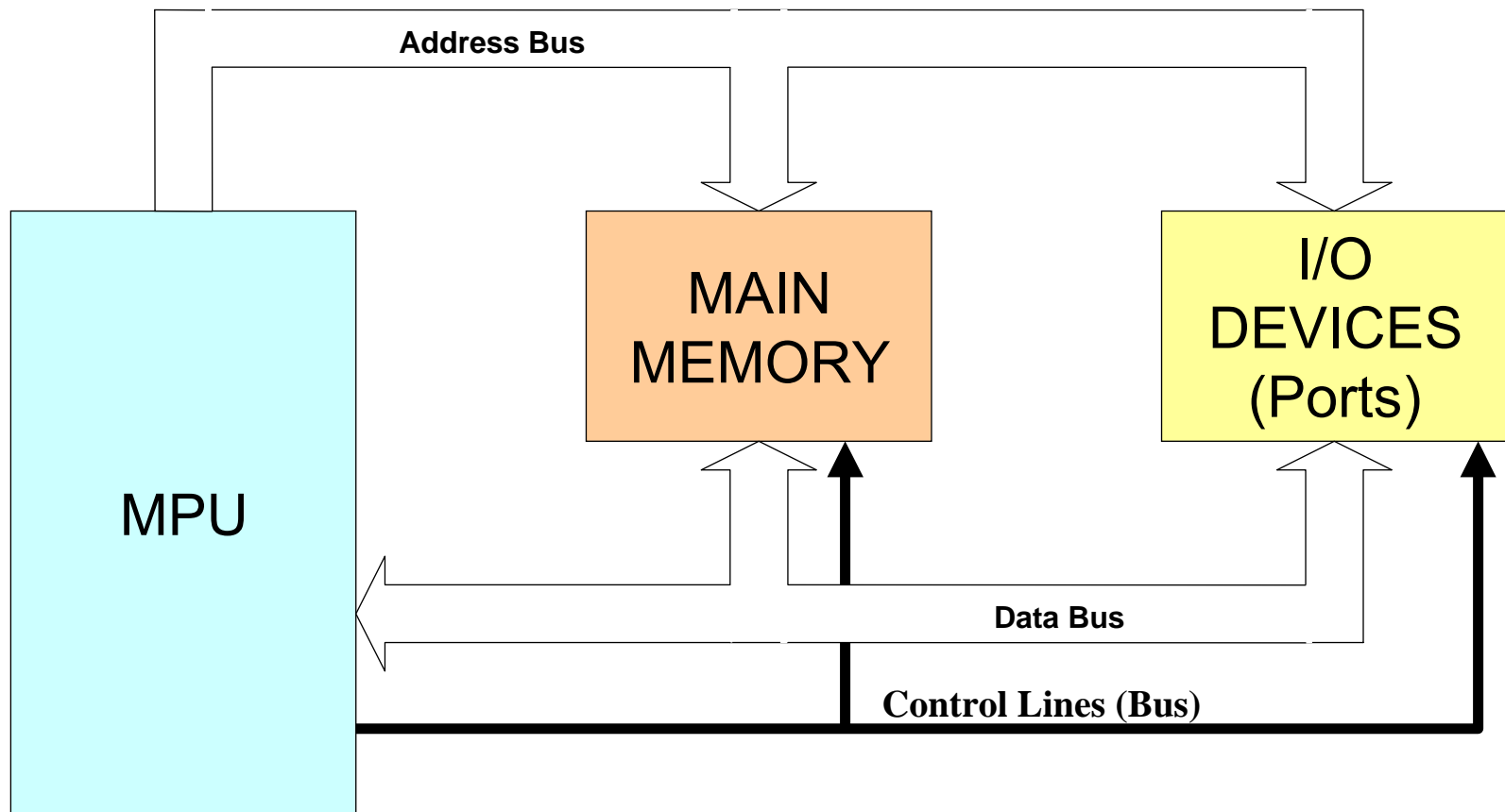
## Von Neumann Architecture



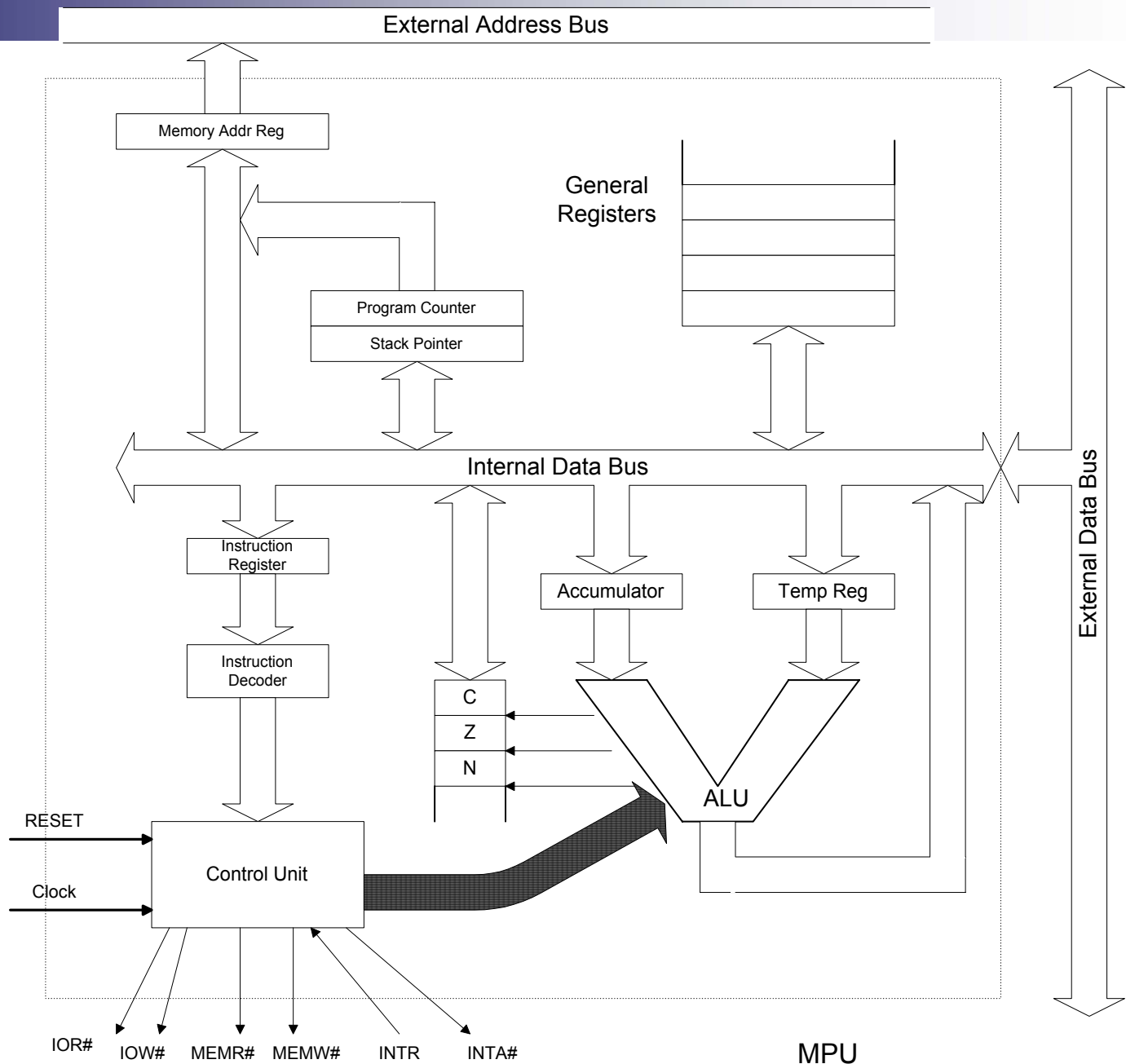
## Harvard Architecture



# Block diagram of a Computer System



# DETAIL OVERVIEW



# MPU vs MCU

## ■ **Micro-processor (MPU, $\mu$ P)**

- ☐ CPU alone
- ☐ may contain some memory
- ☐ classified by data path width 4, 8, 16, 32 or 64 bits

## ■ **Micro-controller (MCU)**

- ☐ microprocessor plus peripherals on a single chip
- ☐ one chip computer system
- ☐ additional peripherals may be interfaced separately
- ☐ Ex: 8051



# The 8086 vs 8088 Microprocessor (1)

## ■ Similarities

### ■ Architecture of the 8088 = 8086:

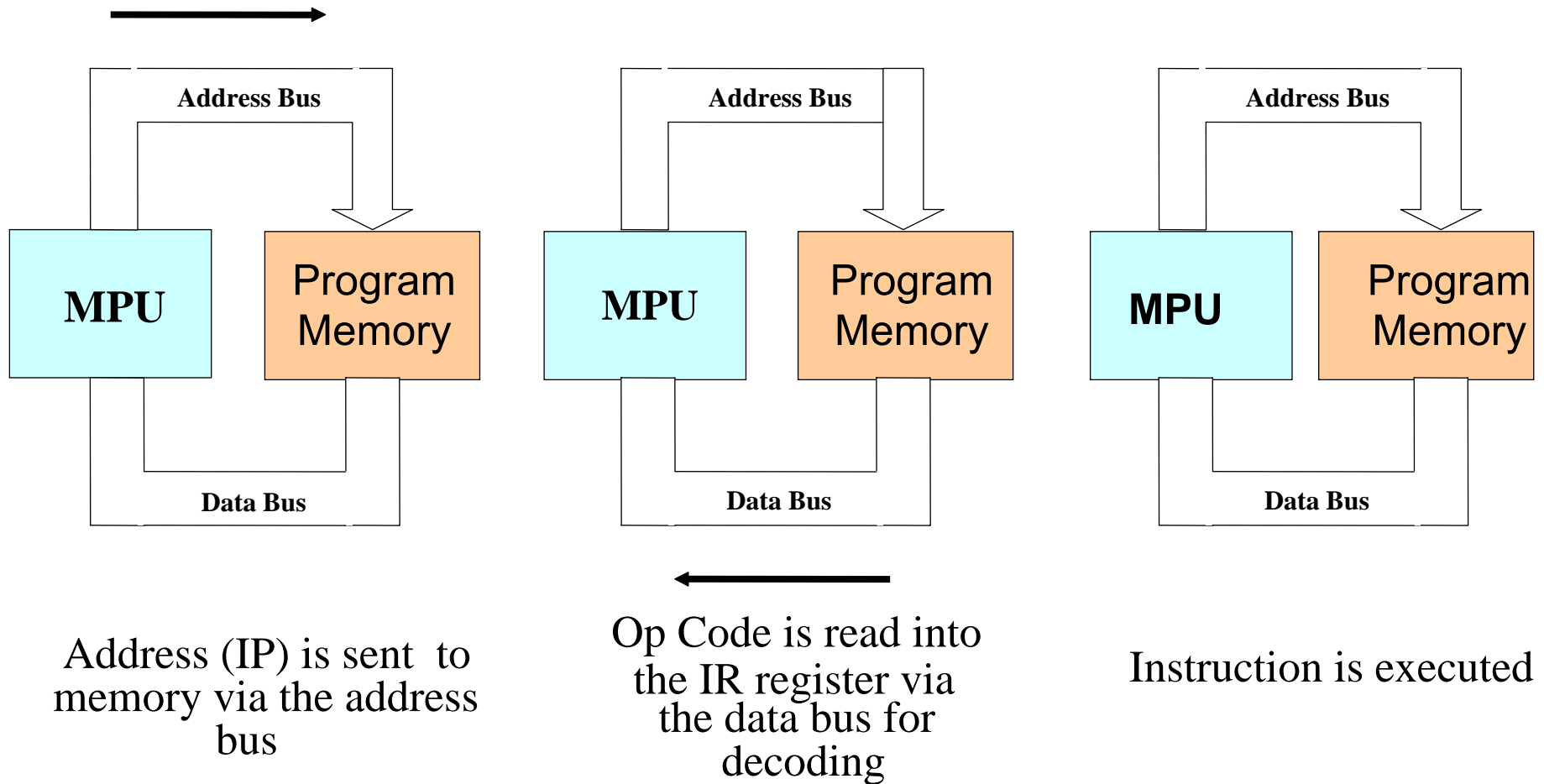
- 16-bit registers, 16-bit internal data bus and 20-bit address bus, (address up to 1 MB of memory).
- 8088 had the same segmented memory addressing as the 8086: the processor could address 64 KB of memory directly, and to address more than 64 KB of memory one of special segment registers had to be updated.

# The 8086 vs 8088 Microprocessor (2)

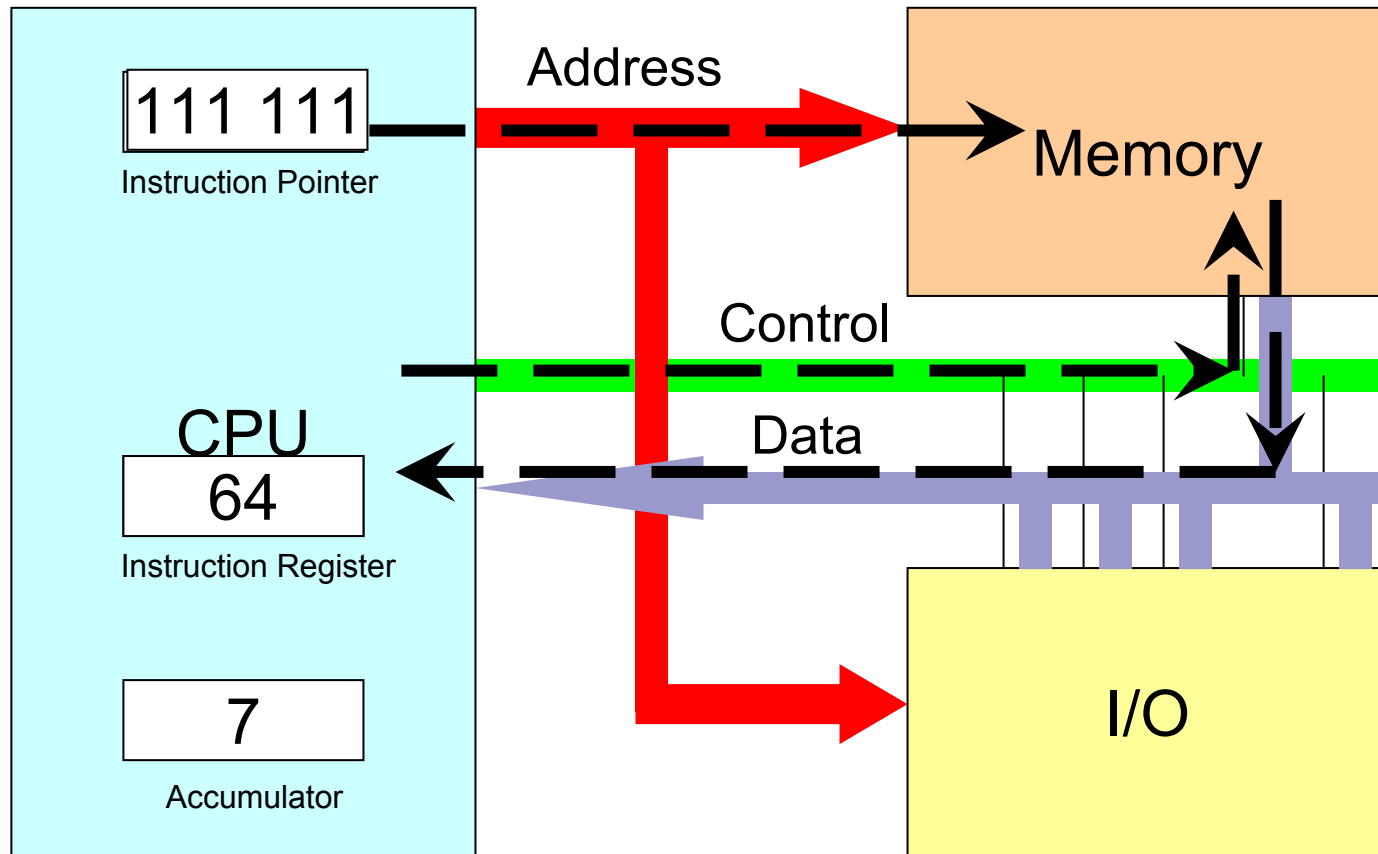
## ■ Differences

- 16-bit data bus in 8086, 8-bit data bus in 8088
- instruction queue size (8088 - 4 bytes long 8086-6 bytes) and prefetching algorithms were changed
  - 8088 used two consecutive bus cycles to write or read 16 bit data instead of one cycle for the 8086.
  - run slower, but on the hardware changes in the 8088 CPU made it compatible with 8080/8085 peripherals.

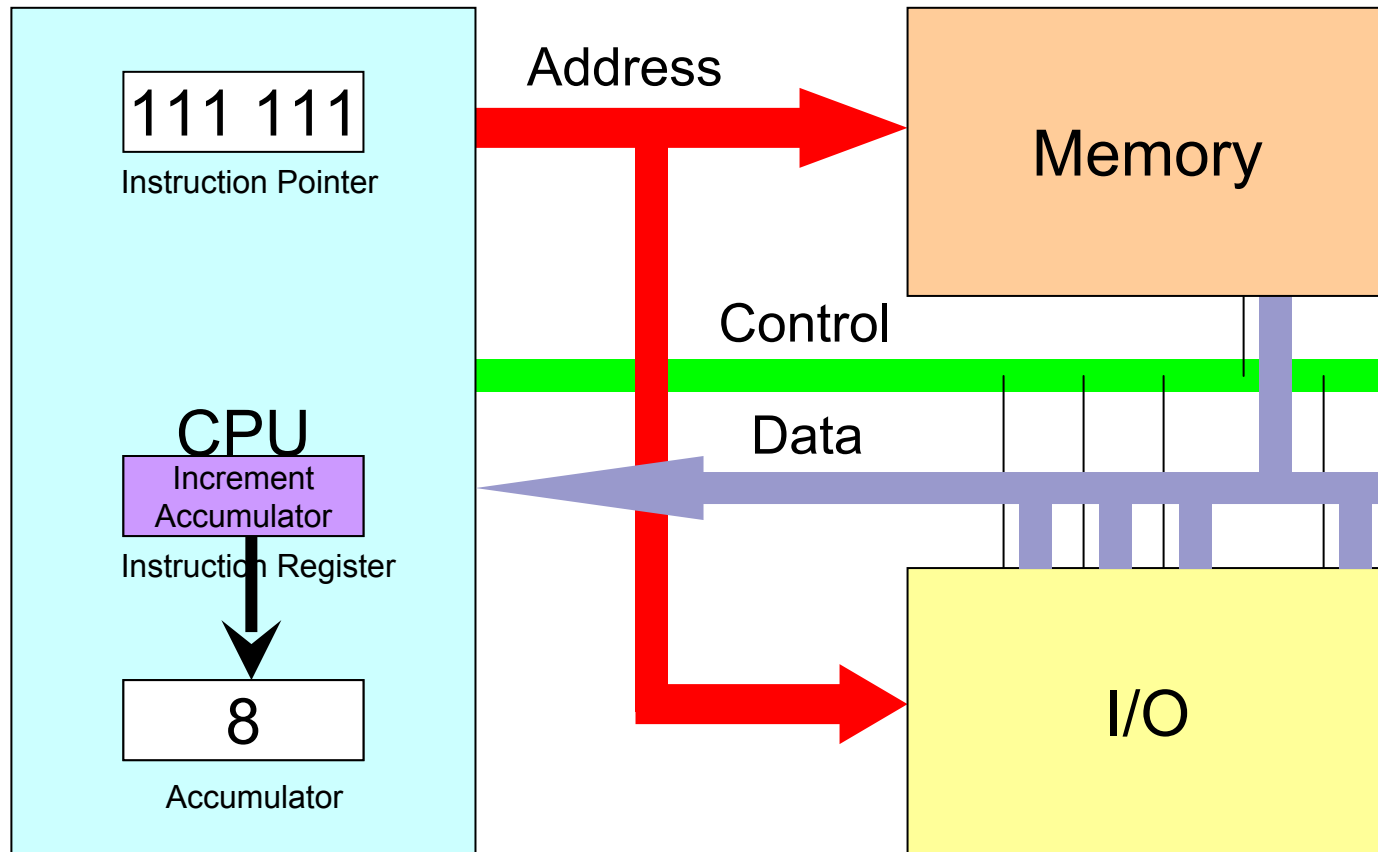
# Fetch-Decode-Execute



# Fetch



# Decode-Execute



# The Sequence

BIU outputs the contents of the IP into the address bus, causing the selected byte or word to be read into the BIU.



IP +1 to prepare for the next instruction fetch



Inside BIU, the instruction is passed to the queue

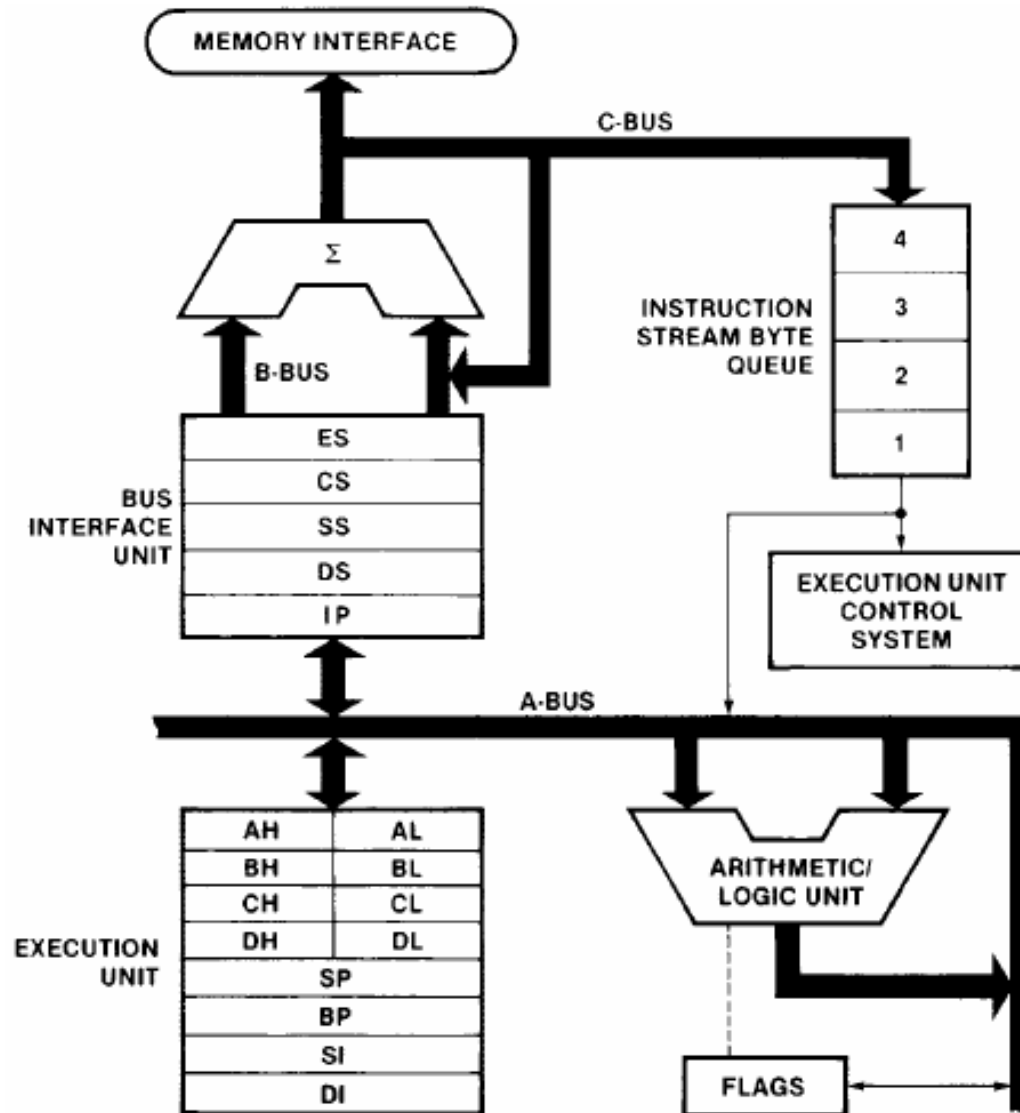


Assuming that the queue is initially empty, the EU immediately draws this instruction from the queue and begins execution



While the EU is executing this instruction, the BIU proceeds to fetch a new instruction.

# 8088 CPU Functional Block (Courtesy Intel Corporation)





# Functions of BIU

- Interface to the external world
- Responsible for all external bus operations.
  - Instruction **fetching** (reading) from primary memory.
  - R/W of data operand from/to primary memory.
  - I/O of data from/to peripheral ports.
  - Address generation for memory reference (Or formation of a 20-bits RAM address from the contents of a segment base registers and and offset register, later).
  - Prefetch instructions for the instructions for the instruction stream queue. It is called pipelined architecture.
- Contents of BIU
  - 4-segment registers (CS, DS, SS and ES, later).
  - An instruction pointer register (IP).
  - Address generation and bus control.
  - Instruction queue (FIFO)-pipeline.

# Functions of EU

- Responsible for **decoding** and **execution** of the instructions.
- Accesses data from the general purpose register
- Check and update control flags (later)
- Commands BUI for memory & I/O operations
- Has the following units:
  - ALU.
  - Status and control flags.
  - General purpose registers (AX, BX, CX, DX, etc).
  - Temporary operand registers.

# Simple operation (1)

Simple CPUs perform one action at a time.

Example instruction sequence

Write to memory Register operation Read from memory
---

CPU:

Fetch	Execute	Write	Fetch	Execute	Fetch	Execute	Read
-------	---------	-------	-------	---------	-------	---------	------

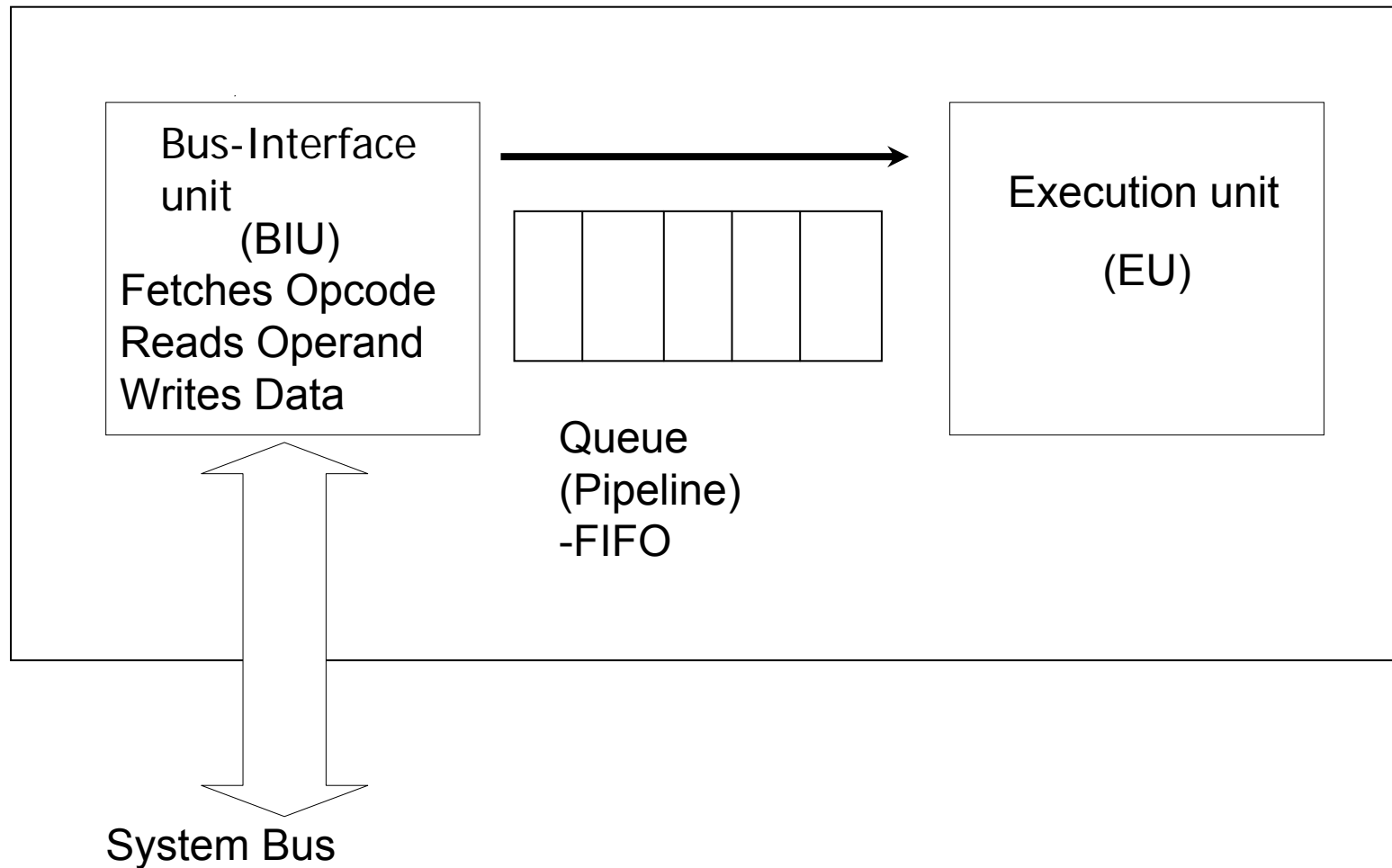
Bus:

Busy		Busy	Busy		Busy		Busy
------	--	------	------	--	------	--	------

# Simple Operation (2)

- Fetching from EXTERNAL MEMORY is SLOW
- The 8086/8 used an instruction queue to speed up performance
- While the processor is decoding and executing an instruction, its bus interface can be reading new instructions, since at that time the bus is not actually in use.

# 8086/8088 Pre-fetching architecture



# 8086/8088 Pre-fetching

The 8086/8088 has a pipelined architecture.

BIU – accesses memory and peripherals

EU – executes fetched instructions

BIU:

Fetch	Fetch	Write	Fetch	Fetch	Read
-------	-------	-------	-------	-------	------

EU:

Idle	Execute	Execute	Idle	Execute	Wait
------	---------	---------	------	---------	------

Bus:

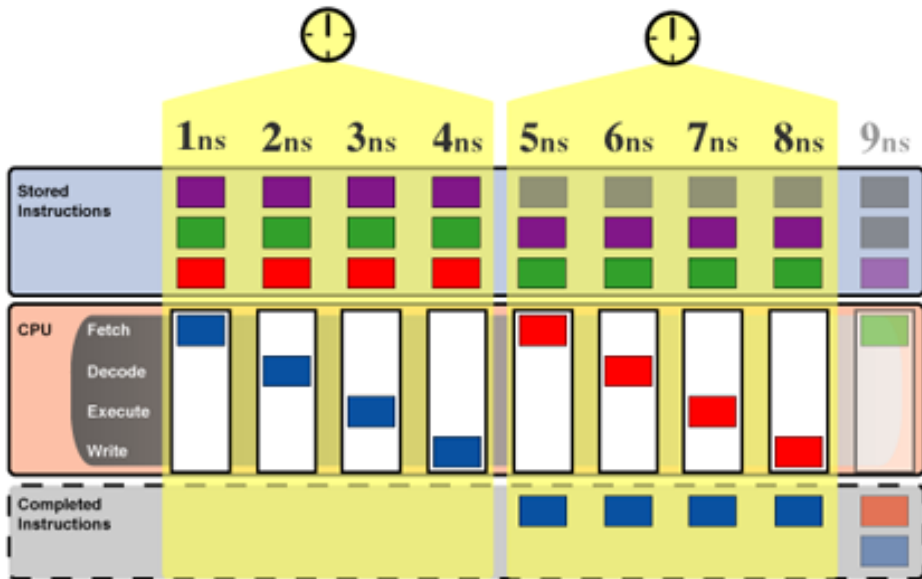
Busy	Busy	Busy	Busy	Busy	Busy
------	------	------	------	------	------

6 cycles instead of 8

Bus is more efficient

BIU pre-fetches instructions bytes whenever EU is not using the bus and stores them in the queue.

# No pipeline vs pipeline





# Physical and Logical Addresses

## ■ Physical Address

- The 20-bit value that uniquely identifies each byte location in the memory

## ■ Logical Address

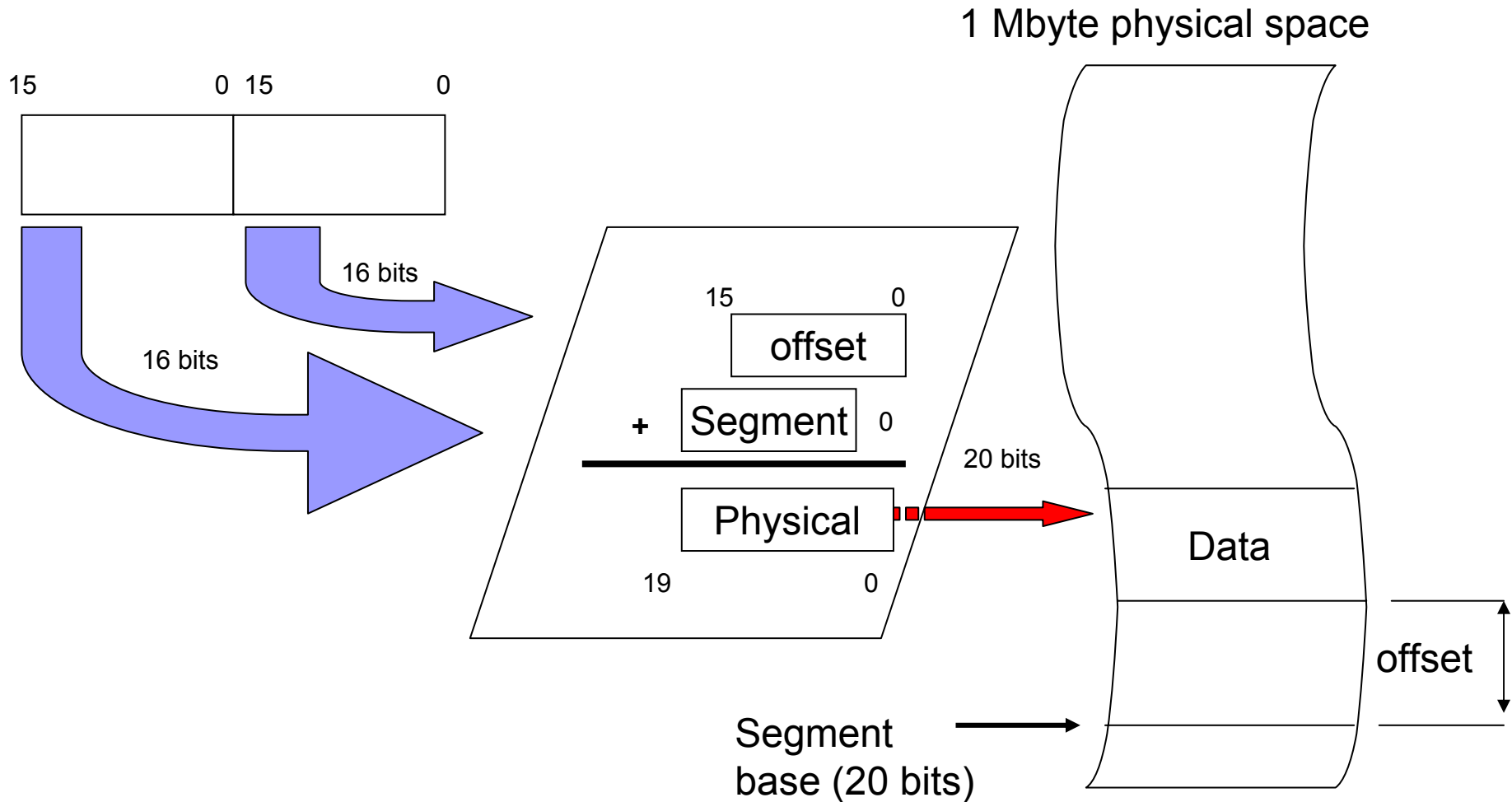
- Allows code to be developed without prior knowledge of where the code is to be located in memory
- Facilitates dynamic management of memory resources
- Consists of a segment base value and offset value

■ 7000:1234 (Logical Address)  
= 8234h (Physical Address)

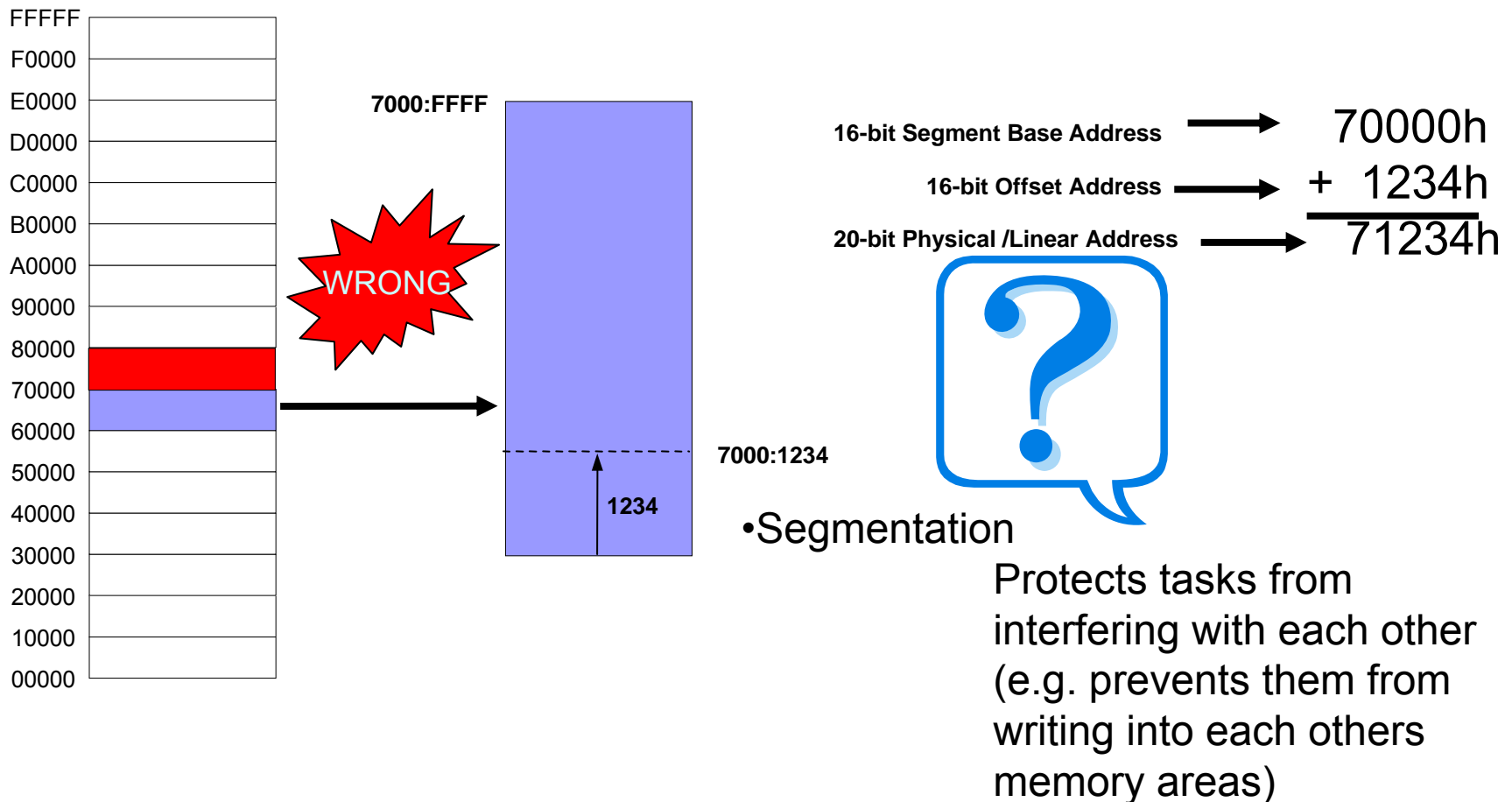
WRONG



# 8088/8086 address generation



# Segmentation

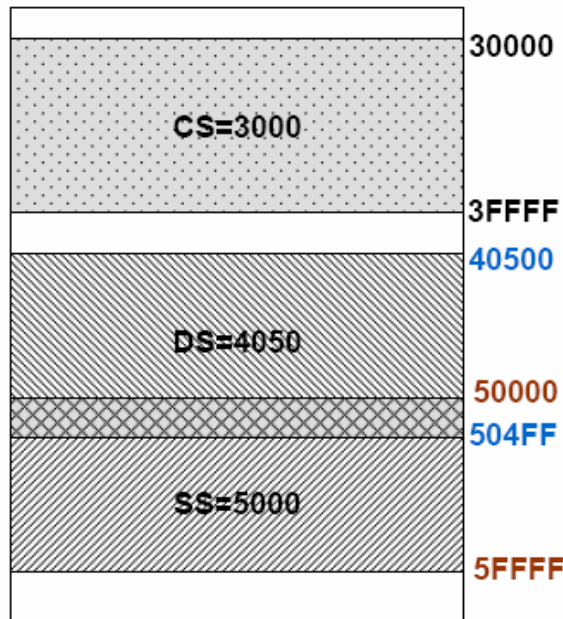


# Restriction value to segment as base

- It must reside on a 16-byte address boundary.

- 0000:0000 = 00000
  - 0001:0000 = 00010 or 0000:0010
  - 0002:0000 = 00020 or 0000:0020
- } 16 bytes  
} 16 bytes

- Segments can be set up to be contiguous, adjacent, disjointed or overlapping



# Segmentation: Pros and cons

- One program can work on several different sets of data. This is done by reloading register DS to a new value.
- Programs that reference logical addresses can be loaded and run anywhere in the memory: relocatable
- Segmented memory introduces extra complexity in both hardware in that memory addresses require two registers.
- They also require complexity in software in that programs are limited to the segment size
- Programs greater than 64 KB can be run on 8086 but the software needed is more complex as it must switch to a new segment.

# Question...

What linear address corresponds to the segment/offset address 05AF:003A?

$$05AF0 + 003A = 05B2A$$

What segment addresses correspond to the linear address 68F50h?

Many different segment-offset addresses can produce the linear address 68F50h. For example:

68F0:0050, 68F5:0000, 68B0:0450, . . .

# Storage organization

All memory in 8086/88 systems are byte-addressable.

Store 12H, 34H, 56H, and 78H in locations 10000H to 10003H.

10003H	78H
10002H	56H
10001H	34H
10000H	12H

In MPU world there are two categories--- Little endian and Big endian

Store 1234H and 5678H in locations 10000H to 10003H.

<b>Big Endian</b>	10003H	78H	10003H	56H	<b>Little Endian</b>
	10002H	56H	10002H	78H	
	10001H	34H	10001H	12H	
	10000H	12H	10000H	34H	

# Little Endian, Big Endian

- Little Endian

- The 'little end' of the number is stored in the least significant byte.

- Big Endian

- The 'big end' of the number is stored in the most significant byte.

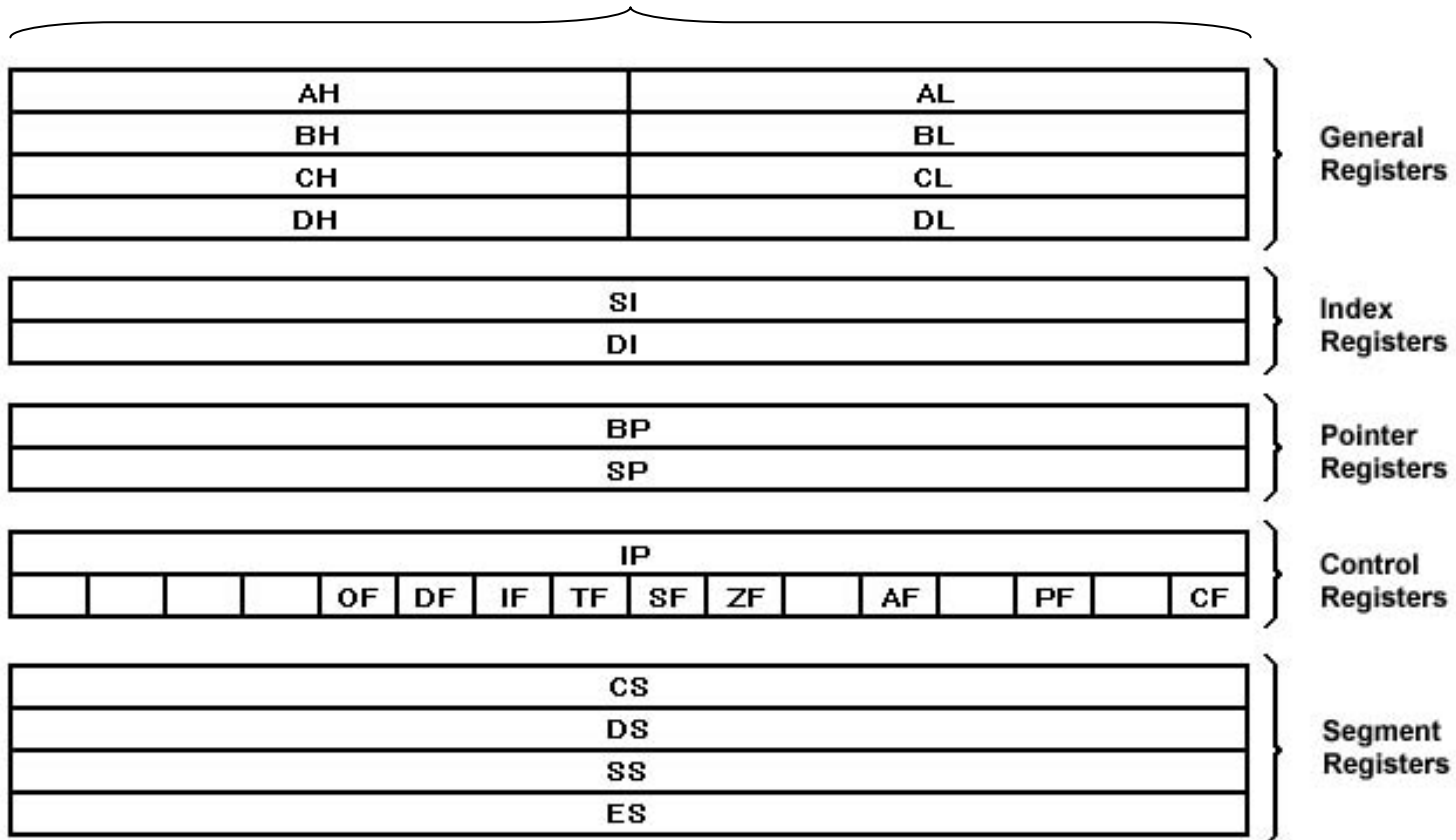
- 8086/8088 uses little endian

- Motorola family uses big endian



# Registers

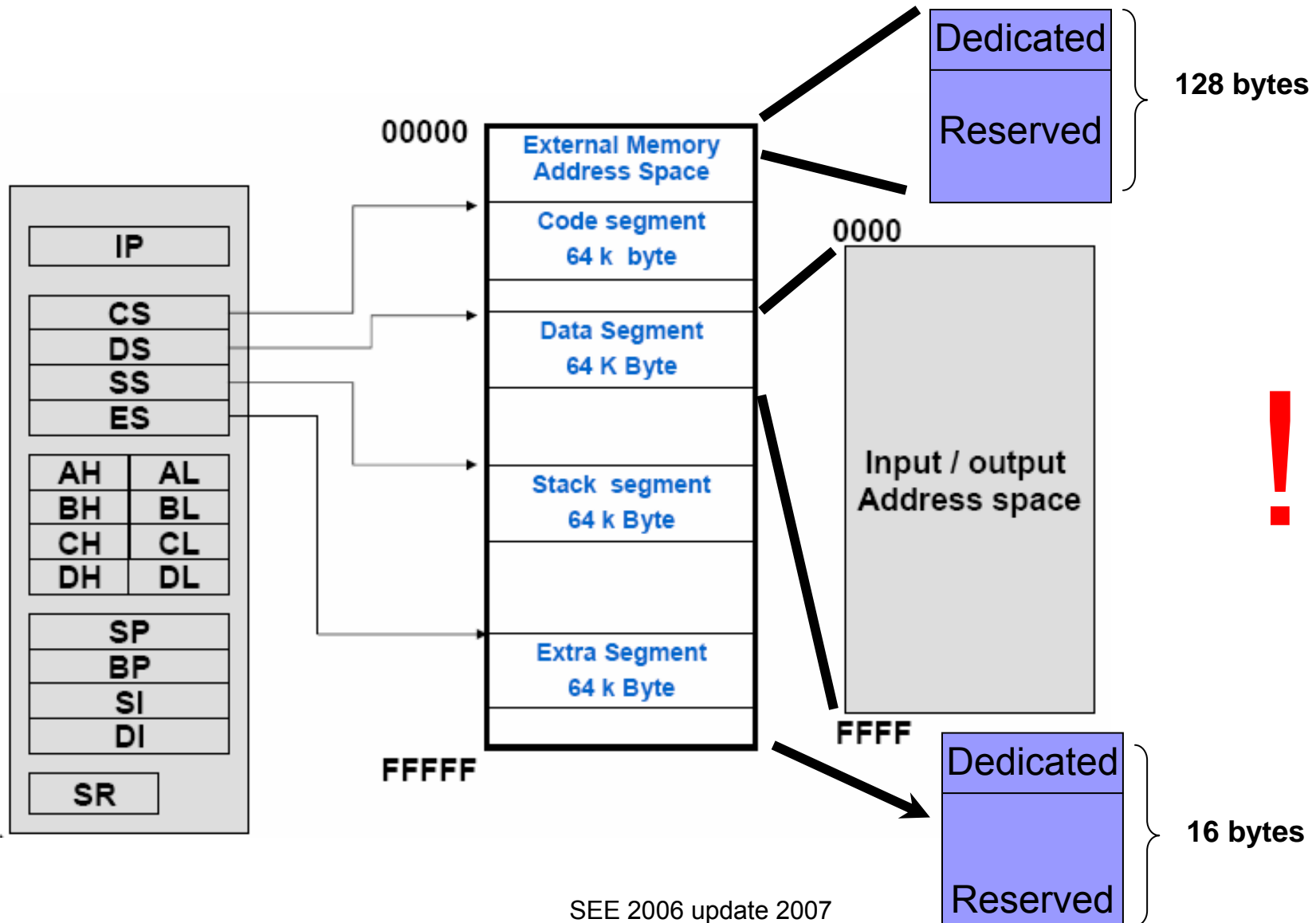
16-bits



# Segments

- 16-bit register containing address of 64 KB segment
- **Only one of the segments can be activated at a time**
- **Code segment (CS)**
  - contains processor instructions (assembly).
- **Stack segment (SS)**
  - For temporary storage of data. By default, the processor assumes that all data referenced by the stack pointer (SP) and base pointer (BP) registers is located in the stack segment.
- **Data segment (DS)**
  - To store data that needs to be processed. By default, the processor assumes that all data referenced by general registers (AX, BX, CX, DX) and index register (SI, DI) is located in the data segment.
- **Extra segment (ES)**
  - Secondary general purpose data area. Defines the area of memory used by some of the string instructions to hold destination data.

# 8088/86 Programming model



# QUIZ...

What is the total memory can be activated each time?

$$4*64KB=256KB$$

# General Purpose Register

- **4-general register with 2 8-bit register (high and low)**
- **Accumulator** register (AL & AH= AX).
  - Used for I/O operations and string manipulation.
- **Base** register (BL & BH=BX).
  - Usually contains a data pointer used for based, based indexed or register indirect addressing.
- **Count** register (CL & CH= CX).
  - Used as a counter in string manipulation and shift/rotate instructions.
- **Data** register (DL & DH= DX).
  - Used as a port number in I/O operations. In integer 32-bit multiply and divide instruction the DX register contains high-order word of the initial or resulting number.

# General Register : Summary

Register	Operation
AX	Word Multiply, word Divide word input output
AL	byte Multiply, byte Divide byte input output, translate, decimal arithmetic
AH	byte Multiply, byte Divide,
BX	<del>Translate</del> Store address information
CX	String operations, loops
CL	Variable shift and rotate
DX	Word multiply, word divide, indirect IO

# String Index

- **Source Index (SI)** - 16-bit register.
  - Used with data segment (DS:SI) and refer to offset locations within DS
  - SI hold the offset address of a data operand to be **READ** from DS.
  - used for indexed, based indexed and register indirect addressing, as well as a source data address in string manipulation instructions.
- **Destination Index (DI)** -16-bit register.
  - Used with data segment (DS:DI) **ES:DI**
  - DI holds the offset address of a data operand to be **WRITTEN** in DS
  - used for indexed, based indexed and register indirect addressing, as well as a destination data address in string manipulation instructions.

# Stack Pointers

## ■ Stack Pointer (SP)

- 16-bit register pointing to program stack.
- Used with SS (SS:SP) register
- Always points to the top of the stack.

## ■ Base Pointer (BP)

- 16-bit register pointing to data in stack segment.
- Used with SS (SS:BP) register
- BP register is usually used for based, based indexed or register indirect addressing (later).
- Commonly used to access parameters that are passed to a subroutine.



# More on stack

- Used as temporary storage
- Used with instructions PUSH & POP
- LIFO
- $\text{PUSH} \rightarrow \text{SP}-2$        $\text{POP} \rightarrow \text{SP}+2$
- All, except the segment registers & SP ,  
can be pushed and popped

# PUSH

- Assume SP=1230H, AX=2107H, DI=1235H, DX=2345H  
Show the contents of the stack as each of the following instructions is executed:  
PUSH AX      PUSH DI      PUSH DX

# POP

- Show the contents of the stack as each of the following instructions is executed

POP AX

POP DI

POP DX

# IP & Flags

## ■ Instruction Pointer (IP)

- 16-bit register which points to next instruction to be executed.
- Contains the offset of the next instructions to be fetched from the CS instead of the actual address
- Every fetch the value IP increment by 2


# Control Flags (1)

Flags <sub>H</sub>								Flags <sub>L</sub>							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	X	X	X	OF	DF	IF	TF	SF	ZF	X	AF	X	PF	X	CF

CF	Carry Flag	high-order bit carry or borrow
PF	Parity Flag	low-order 8 bits contain even number of 1's
AF	Auxiliary Carry	low-order 4 bits carry or borrow of AL
ZF	Zero Flag	result is zero
SF	Sign Flag	equal to high-order bit
TF	Single-step Flag	if set, single-step interrupt occurs
IF	Interrupt-enable	if set, maskable interrupts enabled
DF	Direction Flag	if set, auto-decrement for string instructions
OF	Overflow Flag	signed result too large for destination

# Control flags 2

- Flags is a 16-bit register containing 9 1-bit flags:
- Overflow Flag (OF) - set if the result is too large positive number, or is too small negative number to fit into destination operand.
- Direction Flag (DF) - if set then string manipulation instructions will auto-decrement index registers. If cleared then the index registers will be auto-incremented.
- Interrupt-enable Flag (IF) - setting this bit enables maskable interrupts.
- Single-step Flag (TF) - if set then single-step interrupt will occur after the next instruction.
- Sign Flag (SF) - set if the most significant bit of the result is set.
- Zero Flag (ZF) - set if the result is zero.
- Auxiliary carry Flag (AF) - set if there was a carry from or borrow to bits 0-3 in the AL register.
- Parity Flag (PF) - set if parity (the number of "1" bits) in the low-order byte of the result is even.
- Carry Flag (CF) - set if there was a carry from or borrow to the most significant bit during last result calculation.



Find the contents of flag  
register for the following  
example!!!

# Example 1

```
mov  BH,38H      ; (BH)←38H
add  BH,2FH      ; ADD 2F to (BH) ,now
                  (BH)←67
```

38+2F=67      00111000  
                 00101111  
                 

---

                 01100111

CF=0	PF=0
AF=1	ZF=0
SF=0	



# Example 2

```
MOV  AL,9CH      ; (AL) ← 9CH
MOV  DH,64H      ; (DH) ← 64H
ADD  AL,DH        ; ADD DH to AL ,now
                  AL=0
```

9C+64=100 10011100

01100100

00000000

CF=1

PF=1

AF=1

ZF=1

SF=0

# Example 3

MOV AX,34F5H ; AX=34F5

ADD AX,95EBH ; now AX=CAE0

35F5+95EB=CAE0

0011010111110101

1001010111101011

---

1100101011100000

CF=0

PF=0

AF=1

ZF=0

SF=1

# Example 4

MOV BX,AAAAH ; BX=AAAAH  
ADD BX,55 56H ; now BX=CAE0H

AAAA+5556=100000000

1010101010101010  
0101010101010110  
-----  
0000000000000000

CF=1	PF=1
AF=1	ZF=1
SF=0	

# Example 5

Add the two signed numbers +96 ,+70

MOV AL,60H ; AL=0110 0000 (+96)

MOV BL,46H ; BL=0100 0110 (+60)

ADD AL,BL

0110 0000

0100 0110

1010 0110

CF=0	OF=1	PF=1
AF=0		ZF=0
SF=1		

# Understanding time!!!

## Q.1


Let said a 16-bits data bus. How many electrical lines are there in the address bus?

- A. 16
- B. Unknown



# Answer - B

- The number of lines in the address bus is independent of the number of lines in the data bus.



## Q.2

What is  $10101101111_2$  in hex?

- A.  $AD7_{16}$
- B.  $56F_{16}$

# Answer – B (56F)

- Break it up like this: 101 0110 1111
- So hexadecimal is 56F





### Q.3

What is the largest value for a 16-bit signed binary number?

A. + 32 767

B. + 32 768

# Answer – A (32767)

16-bit signed binary number

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

Sign bit

0 = +ve


1 = -ve

Magnitude bits

$2^{15}$  variations = 32768 numbers

Positive range: 0 – 32 767

Negative range: -1 – -32 768




## Q.4

How much memory can be addressed if a computer has an address bus with 20 lines?

- A. 1 MB
- B. 4 KB

# Answer – A (1MB)

- The number of addressable units of memory is  $2^{20} = 1\,048\,576$ .
- The standard addressable unit is a byte.
- 1 048 576 bytes is 1 MB



Q. 5

ROM is \_\_\_\_\_

- A. volatile
- B. non-volatile

# Answer - B

- Read Only Memory is non-volatile.
- Non-volatile means that the contents of the memory are **not** lost when power is removed.



## Q.6

A data bus is 32-bits wide. How many memory banks are needed?


A. 4

B. 2

# Answer - A

- 4 memory banks, if each one stores data as 8 bits units, are needed for a 32-bit data bus.
  - *Not all memory banks are 8 bits wide!*





## Q.7

A computer stores its instructions and data in separate memory units. This architecture is called\_\_\_\_\_.

- A. von Neumann
- B. Harvard

# Answer - B

- Computers with a Harvard architecture store instructions and data in separate memory units.
  - Most often used with Digital Signal Processing (DSP) microprocessors, and microcontrollers (MCUs)