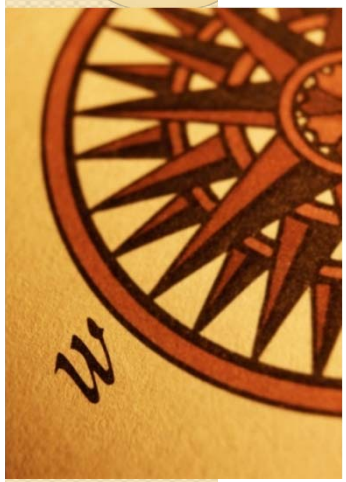
Design and Its application of Microprocessor



The 8088 And 8086 Microprocessors: Programming, Interfacing, Software, Hardware And Applications

Suk-Ju Kang

Dong-A University

kangx@dau.ac.kr

Chapter 2.
Software Architecture of the 8088 and 8086 Microprocessors

In This Chapter, …

- Microarchitecture of the 8088/8086 Microprocessor
- Software Model of the 8088/8086 Microprocessor
- Memory Address Space and Data Organization
- Data Types
- Segment Registers and Memory Segmentation
- Dedicated, Reserved, and General-Used Memory
- Instruction Pointer

In This Chapter, …

- Data Registers
- Pointer and Index Register
- Status Register
- Generating a Memory Address
- The Stack
- Input/Output Address Space

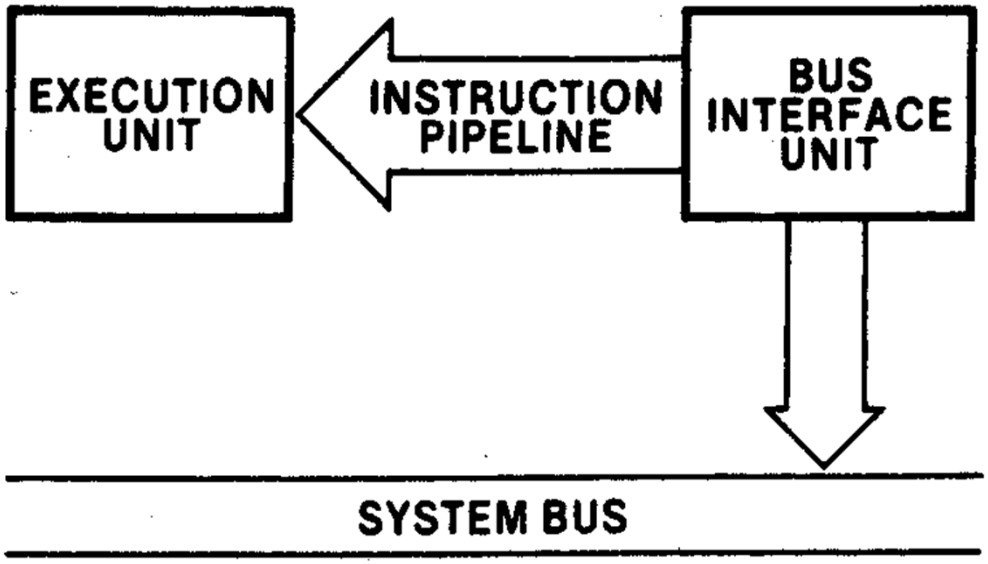
Microarchitecture of the 8088/8086 Microprocessor

- The microarchitecture of a processor is the internal architecture
  → Software and Hardware architecture
- The microarchitecture of the 8088/8086 microprocessor are similar
  - 8088/8086 both employ parallel processing

- 8088/8086 contain two processing unit
  - Bus interface unit (BIU)
  - Execution unit (EU)

Microarchitecture of the 8088/8086 Microprocessor

- Each unit has dedicated functions and both operate at the same time
  → Parallel processing makes the fetch and execution of instructions independent operations



Pipelined Architecture of 8088/8086 microprocessors

Microarchitecture of the 8088/8086 Microprocessor

- The bus interface unit is the path that 8088/8086 connects to external devices
- The system bus includes …
  - 8-bit bidirectional data bus for 8088 (16 bits for the 8086)
  - 20-bit address bus
  - Signals needed to control transfers over the bus

Microarchitecture of the 8088/8086 Microprocessor

- The bus interface unit uses instruction queue

Instruction queue

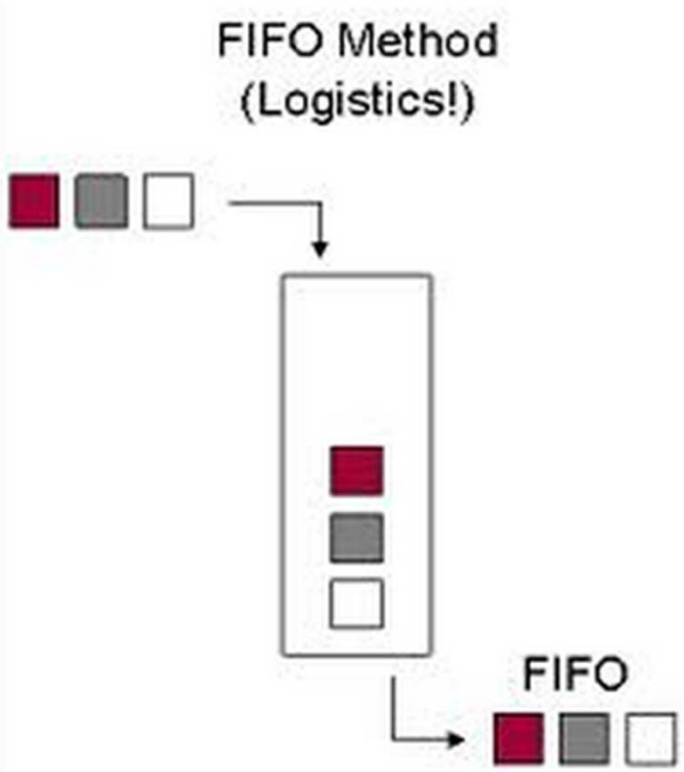
- This queue permits 8088 to prefetch up to 4 bytes (6 bytes for 8086) of instruction code
- Prefetched instructions are held in the First-In-First-Out (FIFO)

FIFO

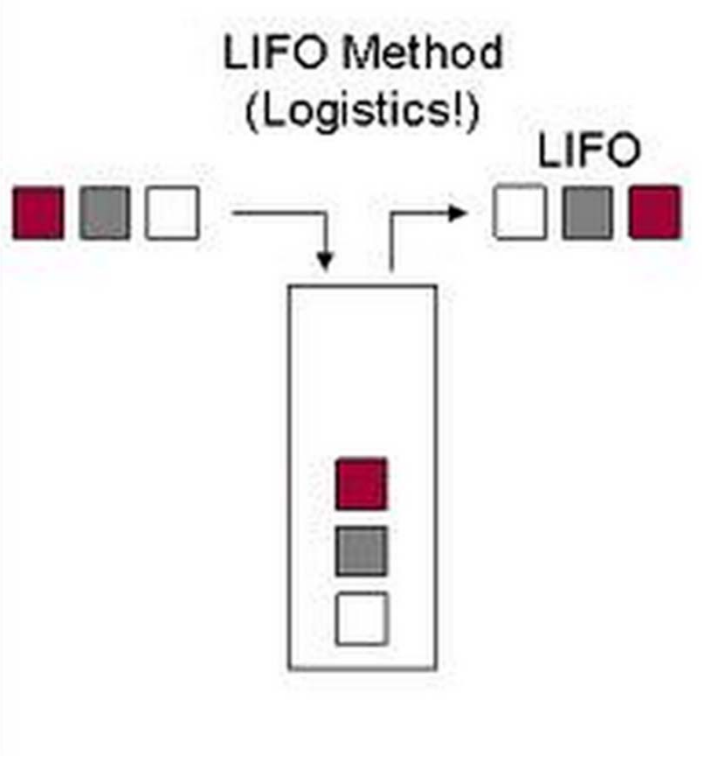
- Process by first-come, first-served (FCFS) behaviour

Microarchitecture of the 8088/8086 Microprocessor

FIFO



LIFO (Last-In-First-Out)



Microarchitecture of the 8088/8086 Microprocessor

- Components in EU
  - Arithmetic logic unit (ALU)
  - Status and control flags
  - General-purpose registers
  - Temporary-operand registers

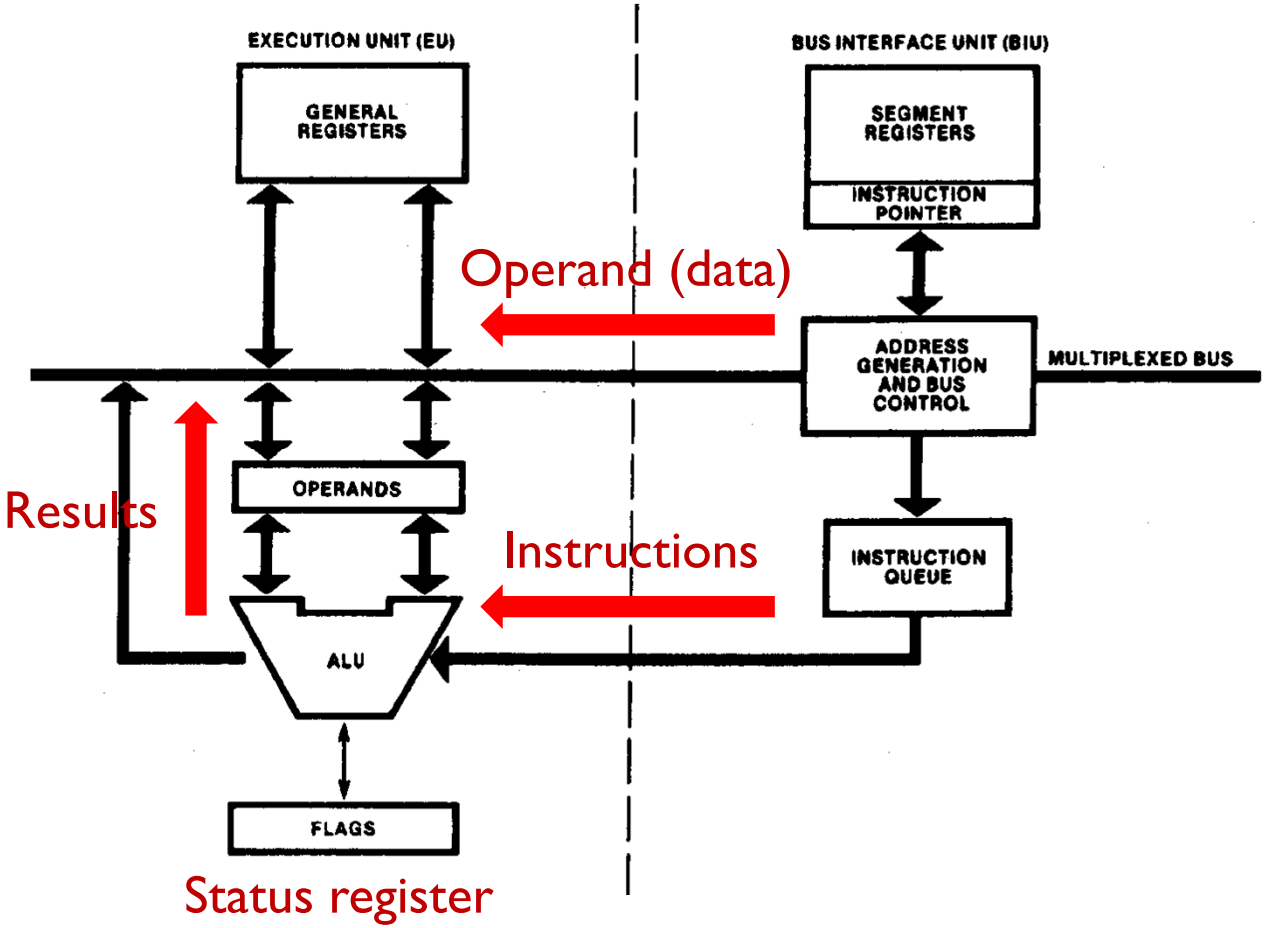
- Components in BIU
  - Segment register
  - The instruction pointer
  - Address generation adder
  - Bus control logic
  - Instruction queue

Microarchitecture of the 8088/8086 Microprocessor

- EU is responsible for decoding and executing instructions
- EU accesses instructions from the output end of the instruction queue and data from the general-purpose registers or memory
- Operations
  - Read one instruction
  - Decode them
  - Generate Data addresses if necessary
  - Passes them to the BIU
  - Request it to perform the read or write operations to memory or I/O
  - Performs the operation specified by the instruction

Microarchitecture of the 8088/8086 Microprocessor

- EU may test the status and control flags and updates these flags based on the results of executing instructions

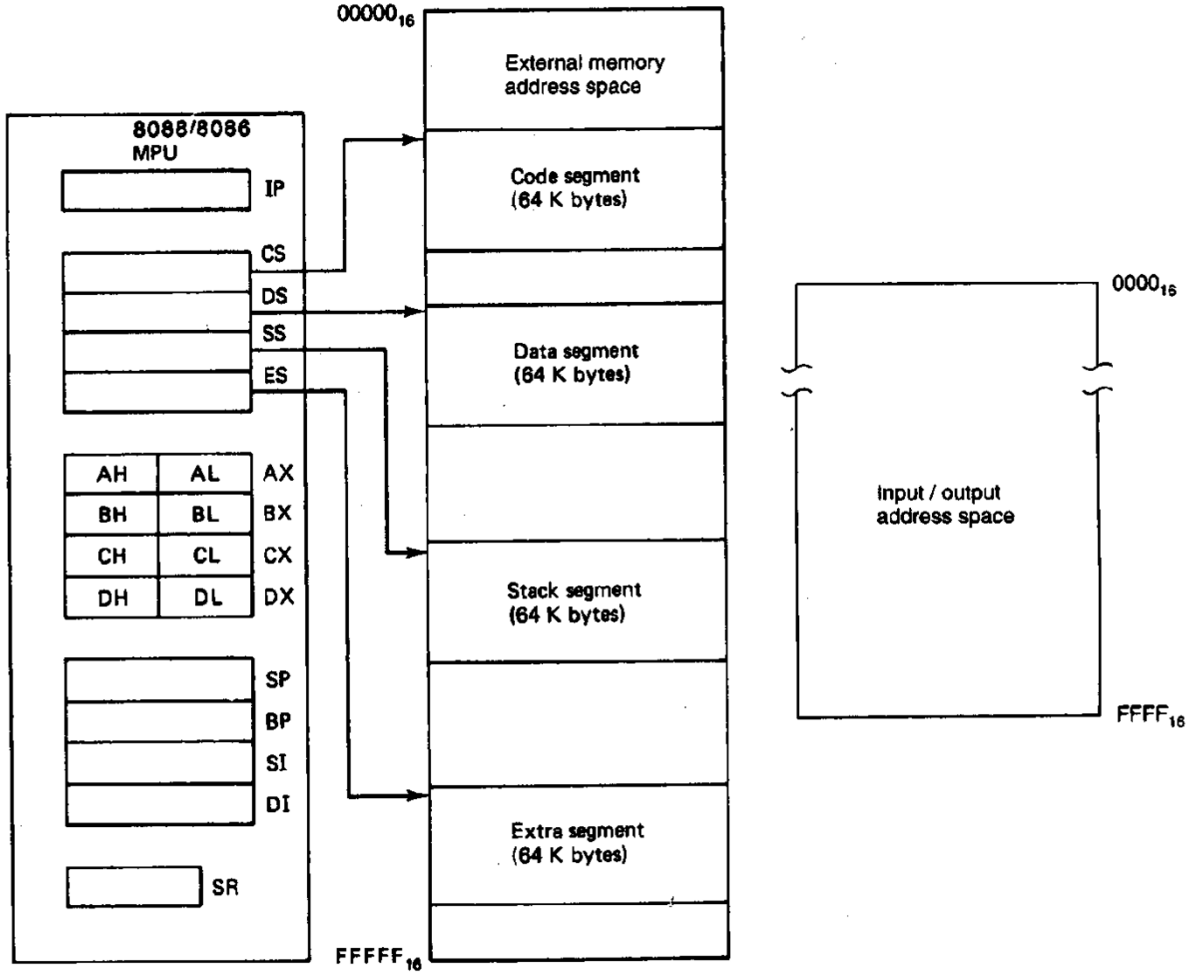


Software Model of the 8088/8086 Microprocessor

- 8088 microprocessor includes 13 16-bit internal registers.
- The instruction pointer : IP
- Four data registers: AX, BX, CX, DX
- Two pointer registers: BP, SP
- Two index registers: SI, DI
- Four segment registers: CS, DS, SS, ES
- Status register, SR, with nine of its bits implemented for status and control flags.
- Memory address space: 1 Mbytes
- I/O address space: 64 Kbytes

Software Model of the 8088/8086 Microprocessor

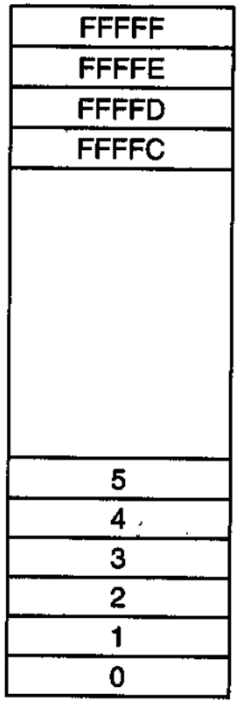
- Software model of the 8088/8086 microprocessor



Memory Address Space and Data Organization

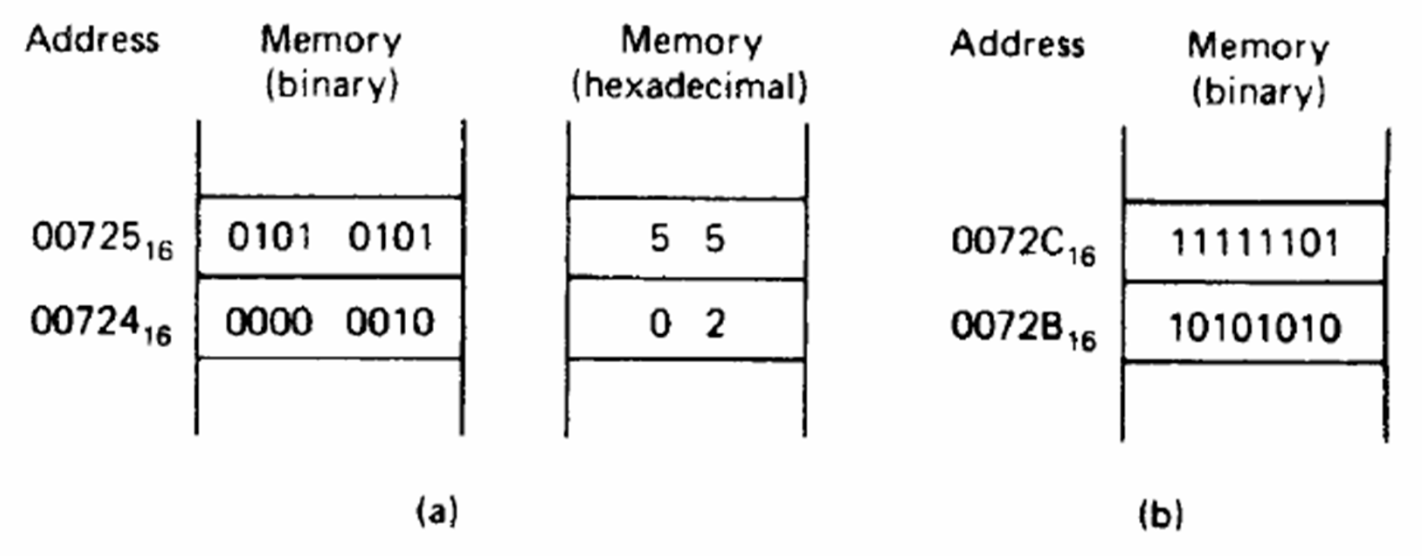
- The 8088 microcomputer supports 1 Mbytes of external memory
- The memory of an 8088-based microcomputer is organized as 8-bit bytes, not as 16-bit words

Memory address space
of 8088/8086 microprocessor



Memory Address Space and Data Organization

- Lower address byte and higher address byte



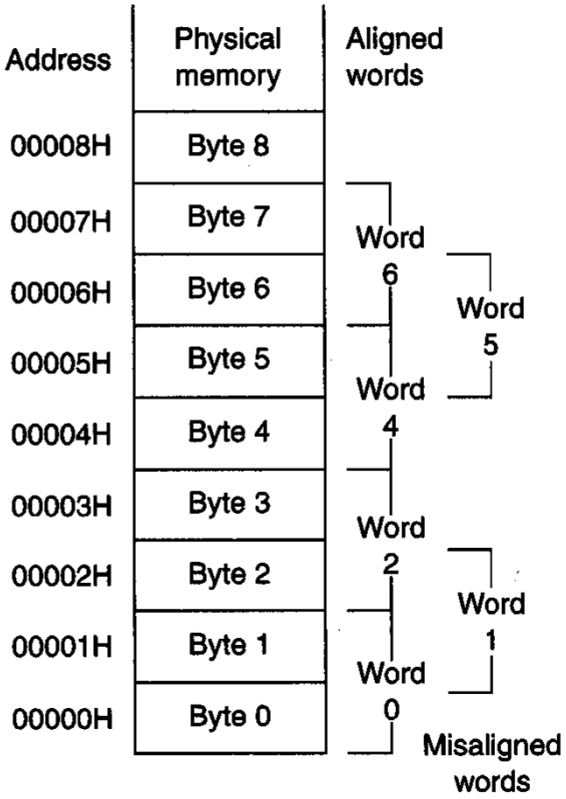
Storing a word of data in memory

The two bytes represent the word

$0101010100000010_2 = 5502_{16}$

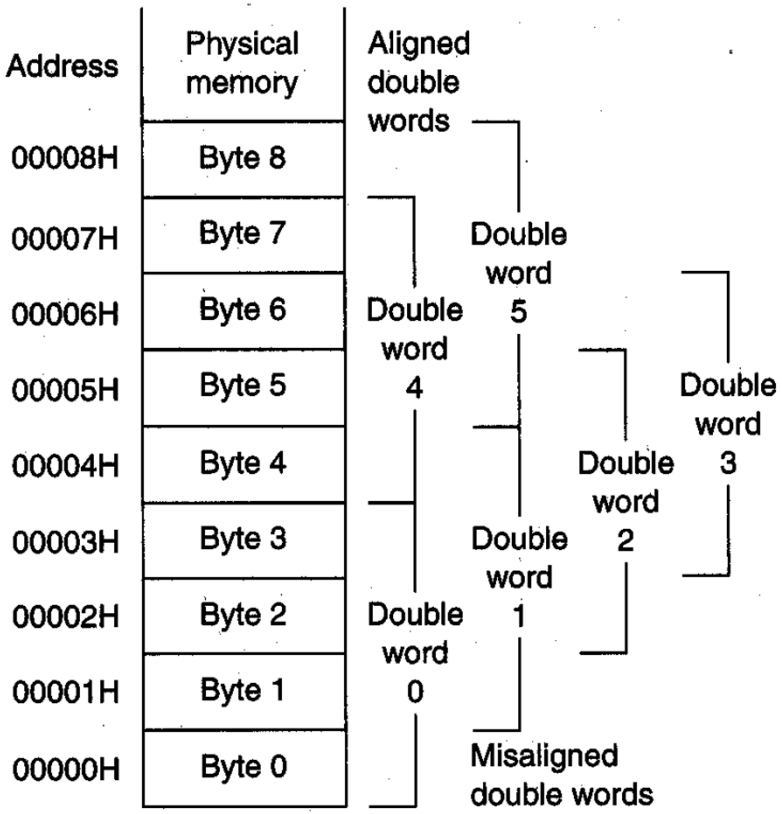
Memory Address Space and Data Organization

- Even- or odd-addressed word
  - If the least significant bit of the address is 0, the word is said to be held at an even-addressed boundary.
- Aligned word or misaligned word



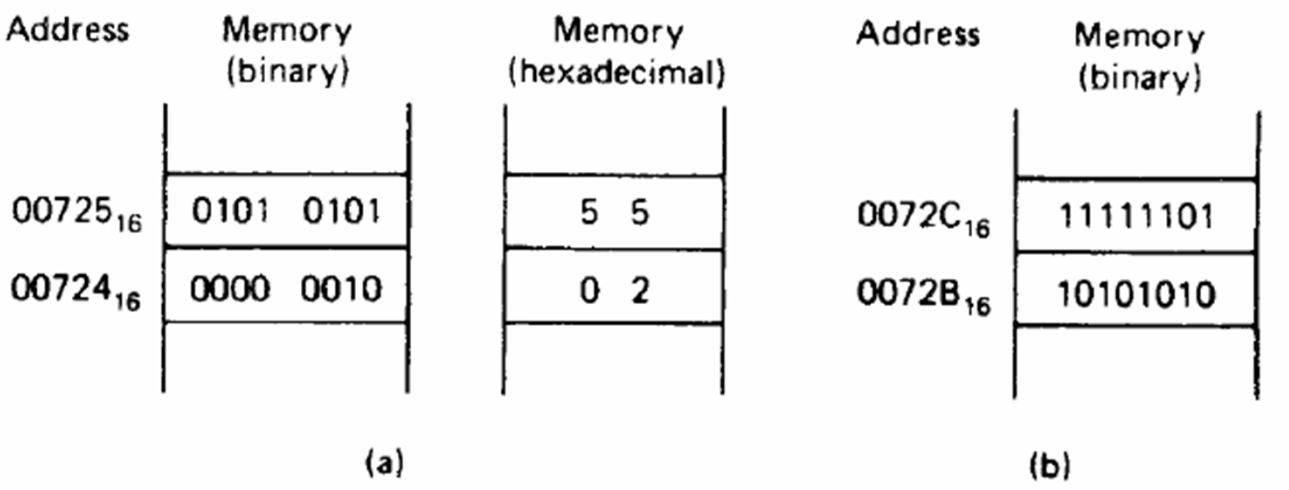
Memory Address Space and Data Organization

- Double word corresponds to four consecutive bytes of data stored in memory



Memory Address Space and Data Organization

- Example:
  - What is the data word shown in the previous figure (b)? (Express the result in hexadecimal form)
  - Is it stored at an even- or odd-addressed word boundary?
  - Is it an aligned or misaligned word of data?



Memory Address Space and Data Organization

Solution:

$11111101_2 = FD_{16} = FDH$

$10101010_2 = AA_{16} = AAH$

Together the two bytes give the word

$1111111010101010_2 = FDAA_{16} = FDAAH$

Expressing the address of the least significant byte in binary form gives

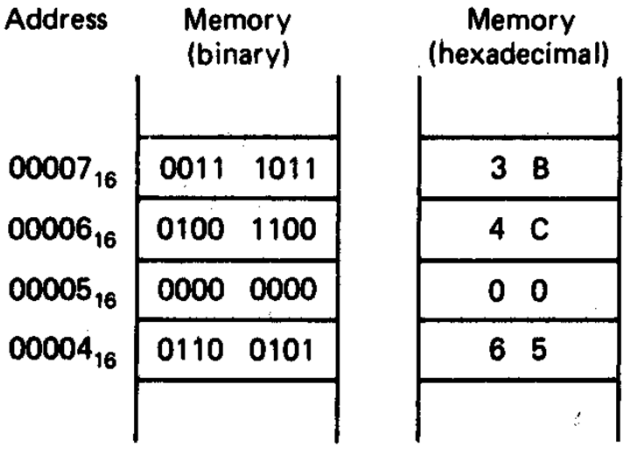
$0072BH = 0072B_{16} =$

00000000011100101011_2

Therefore, it is misaligned word of data

Memory Address Space and Data Organization

- A pointer is a double word
  - The higher address word represents the segment base address
  - The lower address word represents the offset

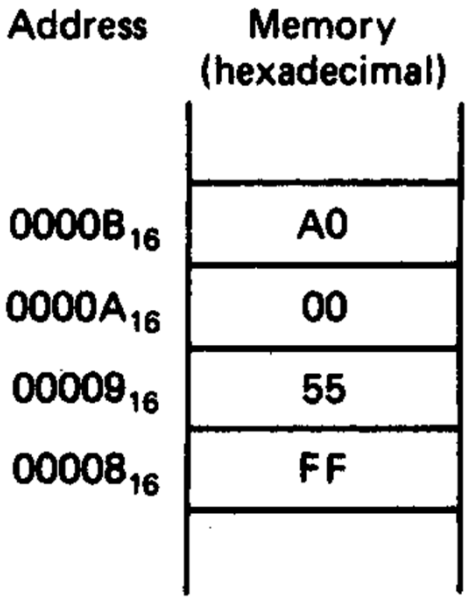


Example:

- Segment base address = $3B4C_{16} = 0011110110100110 0_2$
- Offset = $0065_{16} = 0000000001100101_2$

Memory Address Space and Data Organization

- Example:
  - How should the pointer with segment base address equal to $A000_{16}$ and offset address $55FF_{16}$ be stored at an even-address boundary starting at 00008_{16}?
  - Is the double word aligned or misaligned?

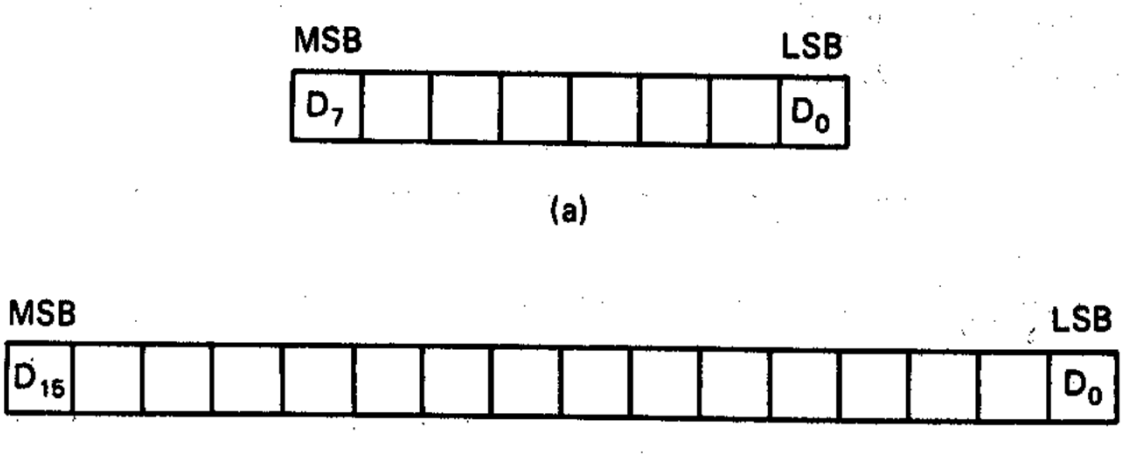


Memory Address Space and Data Organization

- *Solution*:
  - Storage of the two-word pointer requires four consecutive byte locations in memory, starting at address 00008_{16}
  - The least-significant byte of the offset is stored at address 00008_{16} and is shown as FF_{16}
  - The most significant byte of the offset, 55_{16}, is stored at address 00009_{16}
  - These two bytes are followed by the least significant byte of the segment base address, 00_{16}, at address $0000A_{16}$
  - Its most significant byte, $A0_{16}$, at address $0000B_{16}$
  - Since the double word is stored in memory starting at address 00008_{16}, it is aligned

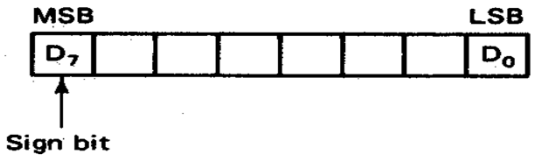
Data Types

- Integer data type
  - Unsigned or signed integer
  - Byte-wide or word-wide integer
- The most significant bit of a signed integer is a sign bit.
  - A zero in this bit position identifies a positive number

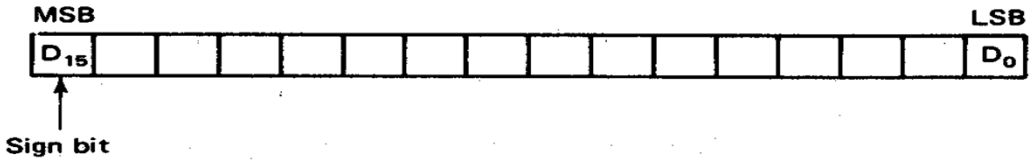


Data Types

- The range of a signed byte integer
  - +127 ~ -128.
- The range of a signed word integer
  - +32767 ~ -32768
- 8088 always expresses negative numbers in 2's complement



(a)



(b)

Data Types

- Example: A signed word integer equals $FEFF_{16}$. What decimal number does it represent?

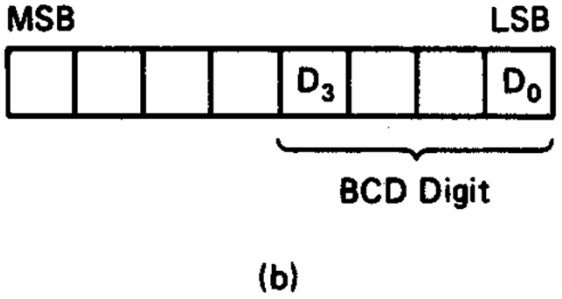
- *Solution:*
  - $FEFF_{16} = 1111111011111111_2$
  - The most significant bit is 1, the number is negative and is in 2's complement form.
  - Converting to its binary equivalent by subtracting 1 from the least significant bit
  - Then complement all bits give:
    - $FEFF_{16} = -0000000100000001_2 = -257$

Data Types

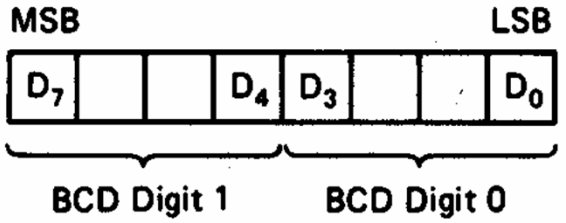
- The 8088 can also process data that is coded as binary-coded decimal (BCD) numbers
- BCD data can be stored in packed or unpacked forms

Decimal	BCD
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

(a)



(b)



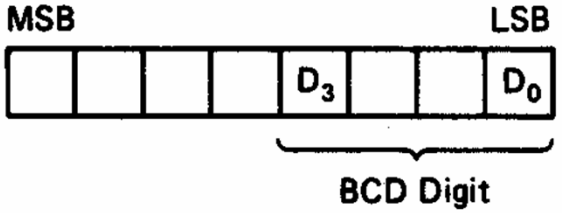
(c)

Data Types

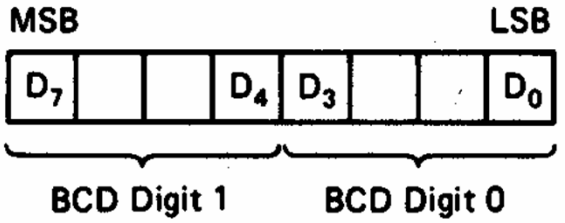
- Unpacked BCD: the upper 4 bits are set to 0
- Packed BCD: 2 BCD numbers are stored (Upper 4 bits represent MSD)

Decimal	BCD
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

(a)



(b)



(c)

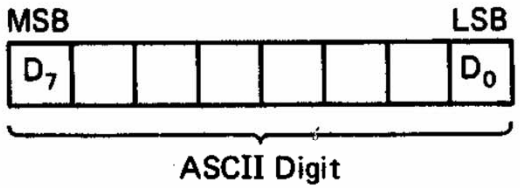
Data Types

- Example: The packed BCD data stored at byte address 01000_{16} equals 10010001_2. What is the two digit decimal number?

- *Solution:*
  - Writing the value 10010001_2 as separate BCD digits gives
  - $10010001_2 = 1001_{BCD}0001_{BCD} = 91_{10}$

Data Types

- ASCII (American Standard Code for Information Interchange)



(b)

	b_7	0	0	0	0	1	1	1	1
	b_6	0	0	1	1	0	0	1	1
	b_5	0	1	0	1	0	1	0	1
$b_4\ b_3\ b_2\ b_1$	H_1 / H_0	0	1	2	3	4	5	6	7
0 0 0 0	0	NUL	DLE	SP	0	@	P	`	p
0 0 0 1	1	SOH	DC1	!	1	A	Q	a	q
0 0 1 0	2	STX	DC2	"	2	B	R	b	r
0 0 1 1	3	ETX	DC3	#	3	C	S	c	s
0 1 0 0	4	EOT	DC4	$	4	D	T	d	t
0 1 0 1	5	ENQ	NAK	%	5	E	U	e	u
0 1 1 0	6	ACK	SYN	&	6	F	V	f	v
0 1 1 1	7	BEL	ETB	'	7	G	W	g	w
1 0 0 0	8	BS	CAN	(	8	H	X	h	x
1 0 0 1	9	HT	EM	)	9	I	Y	i	y
1 0 1 0	A	LF	SUB	*	:	J	Z	j	z
1 0 1 1	B	V	ESC	+	;	K	[	k	{
1 1 0 0	C	FF	FS	,	<	L	\	l	|
1 1 0 1	D	CR	GS	-	=	M	]	m	}
1 1 1 0	E	SO	RS	.	>	N	^	n	~
1 1 1 1	F	SI	US	/	?	O	-	o	DEL

(a)

Data Types

- Example: Byte addresses 01100_{16} through 01104_{16} contain the ASCII data 01000001, 01010011, 01000011, 01001001, and 01001001, respectively. What do the data stand for?

- *Solution:*
  - Using the ASCII table, the data are converted to ASCII code:
  - (01100H) = 01000001_2 = A
  - (01101H) = 01010011_2 = S
  - (01102H) = 01000011_2 = C
  - (01103H) = 01001001_2 = I
  - (01104H) = 01001001_2 = I

Segment Registers and Memory Segmentation

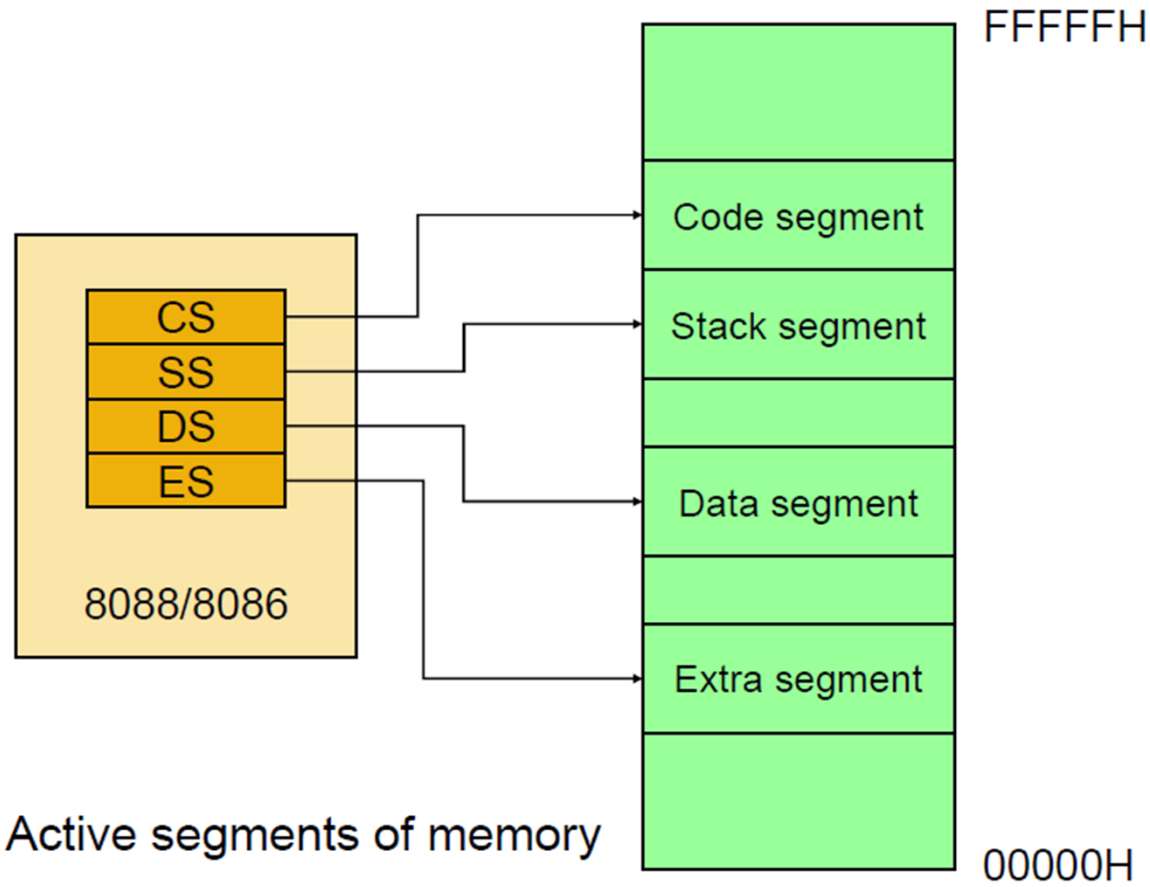
- Even though 8088 has 1 Mbyte address space, not all this memory is active at one time
  - A segment represents an independently addressable unit of memory consisting of 64 Kbyte consecutive bytewide storage locations
  - Each segment is assigned a base address that identifies its starting point

Segment Registers and Memory Segmentation

- Only four segments can be active at a time
  - Code segment
  - Stack segment
  - Data segment
  - Extra segment
- Addresses of the active segments are stored in the four internal segment registers
  - CS, SS, DS, ES

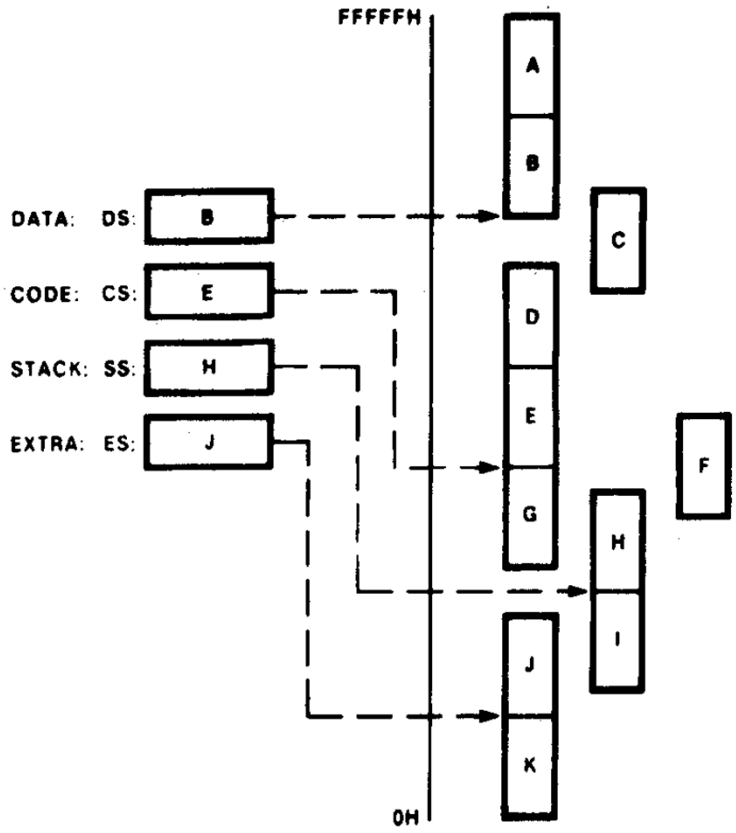
Segment Registers and Memory Segmentation

- Active segments of memory



Segment Registers and Memory Segmentation

- Four segments give a maximum of 256Kbytes of active memory
  - Code segment – 64 Kbytes
  - Stack – 64 Kbytes
  - Data storage – 128 Kbytes



Segment Registers and Memory Segmentation

- Restriction
  - The base address of a segment must reside on a 16 bit address boundary → 1 Unit=16bit
    - Increasing the 16bit value in a segment register by 1 actually increases the corresponding memory address by 16
  - Examples of valid base address
    - $00000_{16}, 00010_{16}, 00020_{16}$
  - User accessible segments can be set up to be contiguous, adjacent, disjointed, or even overlapping

Dedicated, Reserved, and General-Used Memory

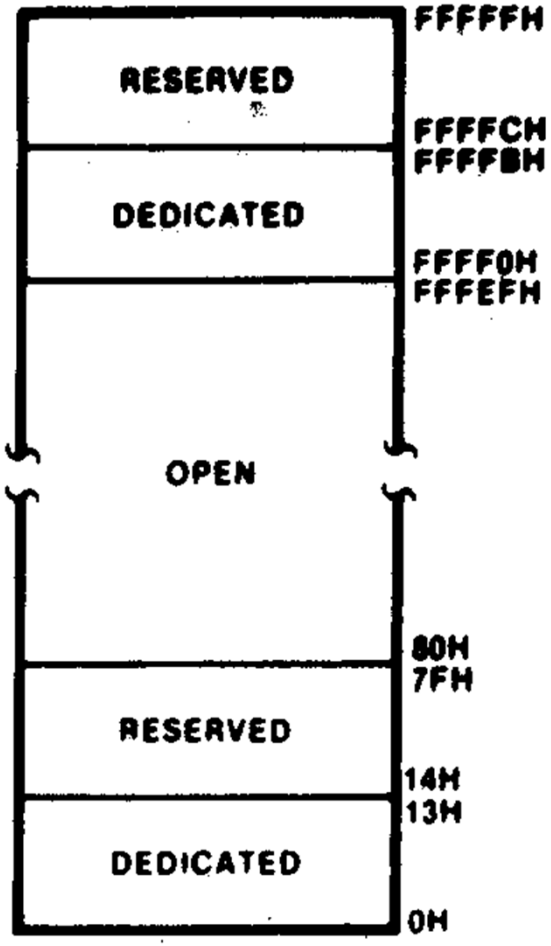
- Dedicated memory (00000_{16} ~ 00013_{16}) are used for storage of the pointers to 8088's internal interrupt service routines and exceptions
- Reserved memory (00014_{16} ~ $0007F_{16}$) are used for storage of the pointers to user-defined interrupts
- These 128-byte dedicated and reserved memory can contain 32 interrupt pointers

Dedicated, Reserved, and General-Used Memory

- General-use memory (00080_{16} ~ $FFFEF_{16}$) stores data or instructions of the program.
- Dedicated memory ($FFFF0_{16}$ ~ $FFFFB_{16}$) are used for hardware reset jump instruction
- Reserved memory ($FFFFC_{16}$ ~ $FFFFF_{16}$) are reserved for use with future products and should not be used

Dedicated, Reserved, and General-Used Memory

- 8088/8086 address space



Instruction Pointer

- Instruction pointer (IP) identifies the location of the next word of instruction code to be fetched from the current code segment of memory
- IP is 16 bits in length
- IP is similar to program counter (PC)
  - But it contains the offset of the next word of instruction code instead of its actual address
  - IP: 16 bits, Actual memory address: 20 bits
- PC is incremented after fetching an instruction
  - It holds the actual memory address of the next instruction that would be executed

Instruction Pointer

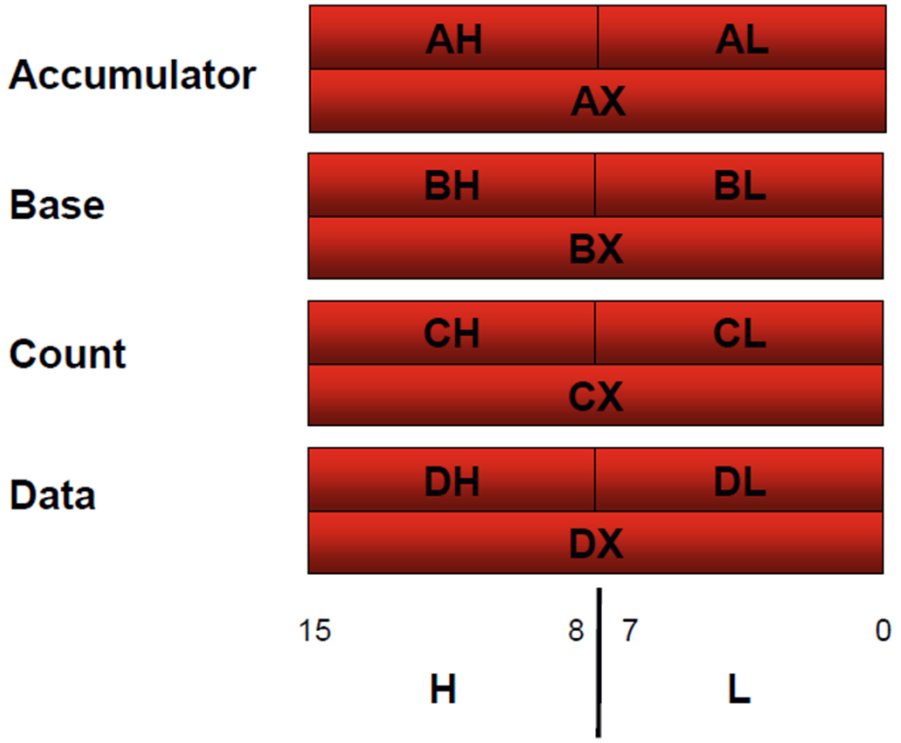
- Offset in IP is combined with the current value in code segment (CS) to generate the address of the instruction code
- During normal operation, the 8088 fetches instructions from the code segment of memory, stores them in its instruction queue, and executes them one after the other

Data Registers

- Data registers are used for temporary storage of frequently used intermediate results
- Contents of the data registers can be read, loaded, or modified through software
- Advantage of storing data in internal registers instead of memory during processing is that they can be accessed much faster
- The four data registers are:
  - Accumulator register, A
  - Base register, B
  - Counter register, C
  - Data register, D

Data Registers

- Each register can be accessed either as a whole (16 bits) for word data or as 8-bit data for byte-wide operation



General-purpose data registers of 8088 microprocessor

Data Registers

- Dedicated register functions

Register	Operations
AX	Word multiply, word divide, word I/O
AL	Byte multiply, byte divide, byte I/O, translate, decimal arithmetic
AH	Byte multiply, byte divide
BX	Translate
CX	String operations, loops
CL	Variable shift and rotate
DX	Word multiply, word divide, indirect I/O

Dedicated register functions

Pointer and Index Register

- The pointer registers and index registers are used to store offset addresses
- Values held in the index registers are used to reference data relative to data segment or extra segment
- The pointer registers are used to store offset addresses of memory location relative to the stack segment register

Pointer and Index Register

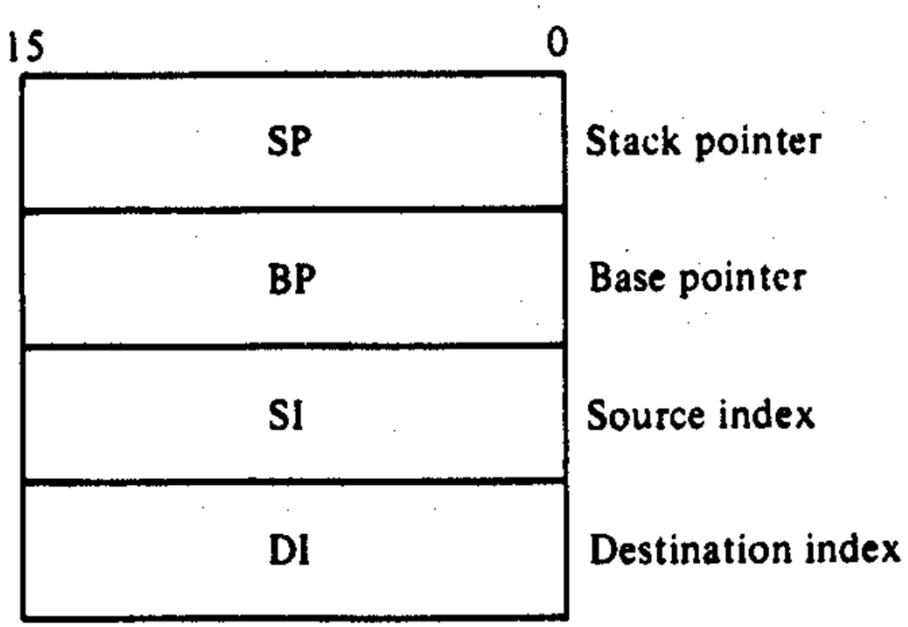
- Combining SP with the value in in SS (SS:SP) results in a 20-bit address that points to the top of the stack (TOS)
- BP is used to access data within the stack segment of memory
  - It is commonly used to reference subroutine parameters

Pointer and Index Register

- The index register are used to hold offset addresses for instructions that access data in the data segment.
- The source index register (SI) is used for a source operand, and the destination index (DI) is used for a destination operand.
- The four registers must always be used for 16-bit operations

Pointer and Index Register

- Pointer and index registers

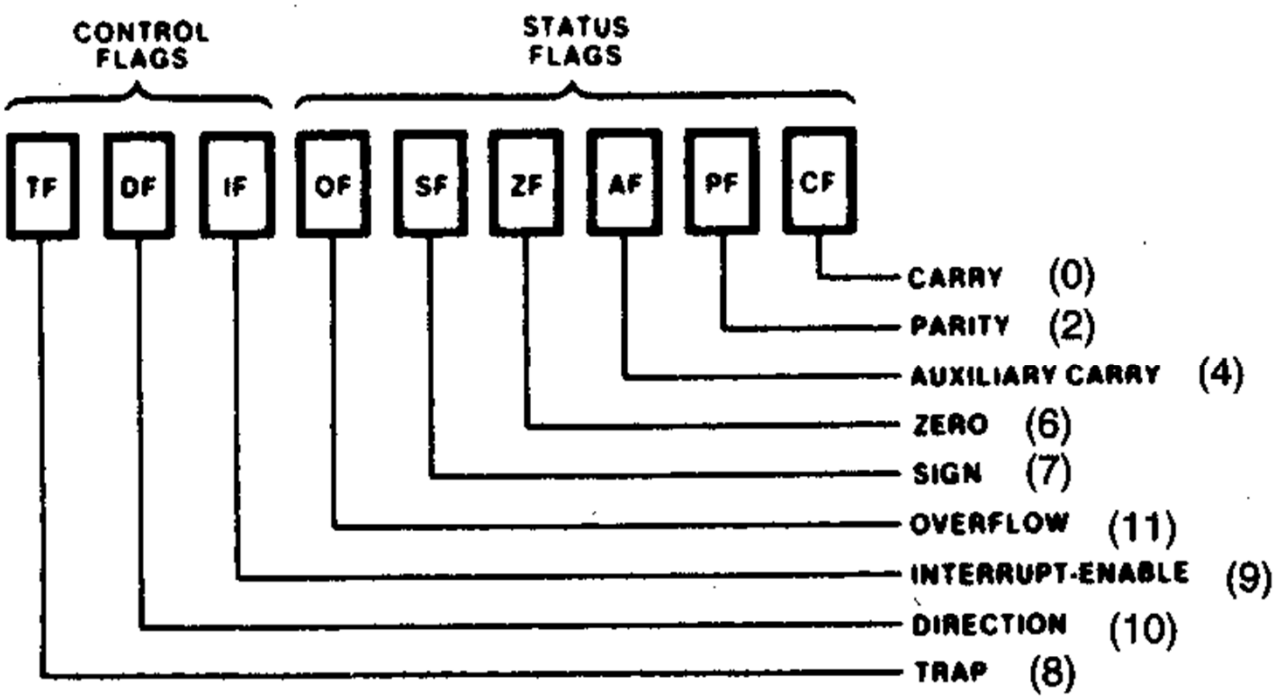


Status Register

- The status register, also called the flags register, indicate conditions that are produced as the result of executing an instruction
- Only 9 bits of the register are implemented
  - 6 bits represent status flags
  - 3 bits represent control flags
- 8088 provides instructions within its instruction set that are able to use these flags to alter the sequence in which the program is executed

Status Register

- Status and control flags



Status Register

- Status flags indicate current processor status

CF	Carry Flag	Carry-out/Borrow-in
OF	Overflow Flag	Overflow (Out of range)
ZF	Zero Flag	Zero Result
SF	Sign Flag	Negative Result
PF	Parity Flag	Even Number of "1" bits
AF	Auxiliary Carry	Carry-out/Borrow-in for BCD Arithmetic

Status Register

- Parity bit?
  - A bit that is added to ensure that the number of bits with the value one in a set of bits is even or odd
  - Parity bits are used as the simplest form of error detecting code

7 bits of data (count of 1 bits)	8 bits including parity	
	even	odd
0000000 (0)	**0**0000000 (0)	**1**0000000 (1)
1010001 (3)	**1**1010001 (4)	**0**1010001 (3)
1101001 (4)	**0**1101001 (4)	**1**1101001 (5)
1111111 (7)	**1**1111111 (8)	**0**1111111 (7)

Status Register

- Status flags indicate current processor status

DF	Direction Flag	Increment or Decrement Low address ↔ High address
IF	Interrupt Flag	Interrupt Request
TF	Trap Flag	Single-step mode

- Single-step mode
  - It executes an instruction and then jumps to a special service routine that may determine the effect of executing the instruction

Generating a Memory Address

- Logical address in the 8088 microcomputer system is described by a segment base and an offset
- Physical address that is used to access memory are 20 bits in length
- Generation of the physical address
  - Combining a 16-bit offset value that is located in the instruction pointer, a base pointer, an index register, or a pointer register and a 16-bit segment base value that is located in one of the segment register

Generating a Memory Address

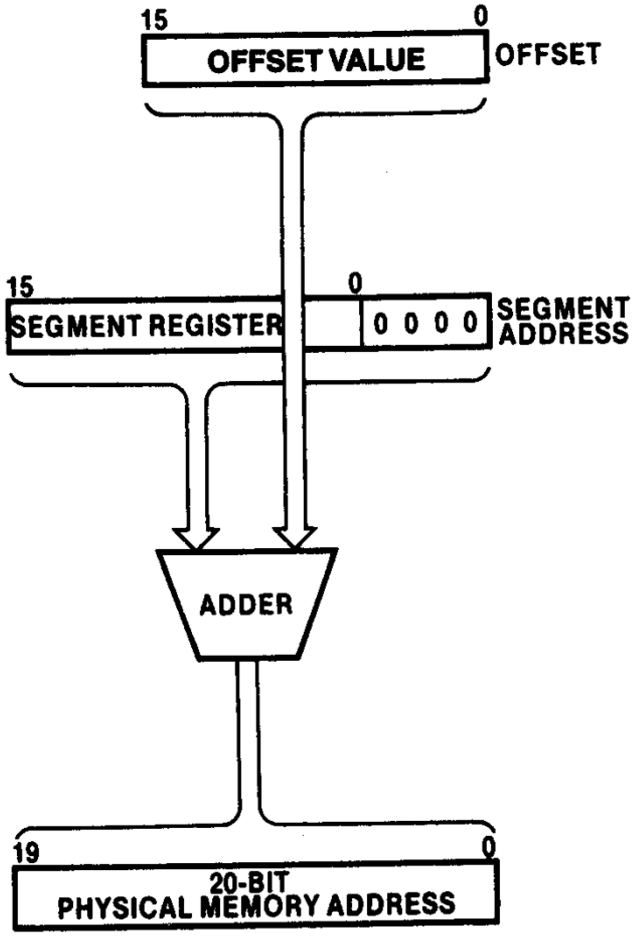
- Source of the offset value depends on which type of memory is taking place
  - Base pointer (BP) register
  - Stack pointer (SP) register
  - Base (BX) register
  - Source index (SI) register
  - Destination index (DI) register
  - Instruction pointer (IP)

Generating a Memory Address

- Segment base value resides in one of the segment registers
  - CS, DS, SS, ES
- Example: Instruction acquisition
  - Source of the offset value: Instruction pointer (IP)
  - Source of the segment base value: Code segment (CS) register
  - Physical address: CS:IP

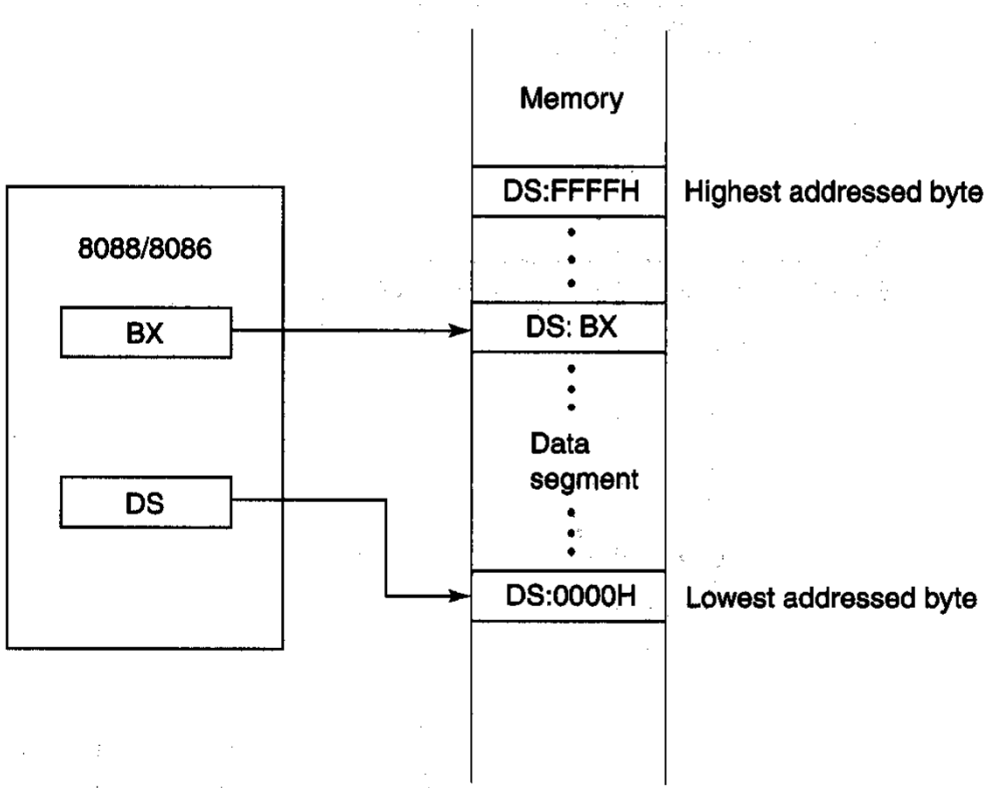
Generating a Memory Address

- Generating a physical address



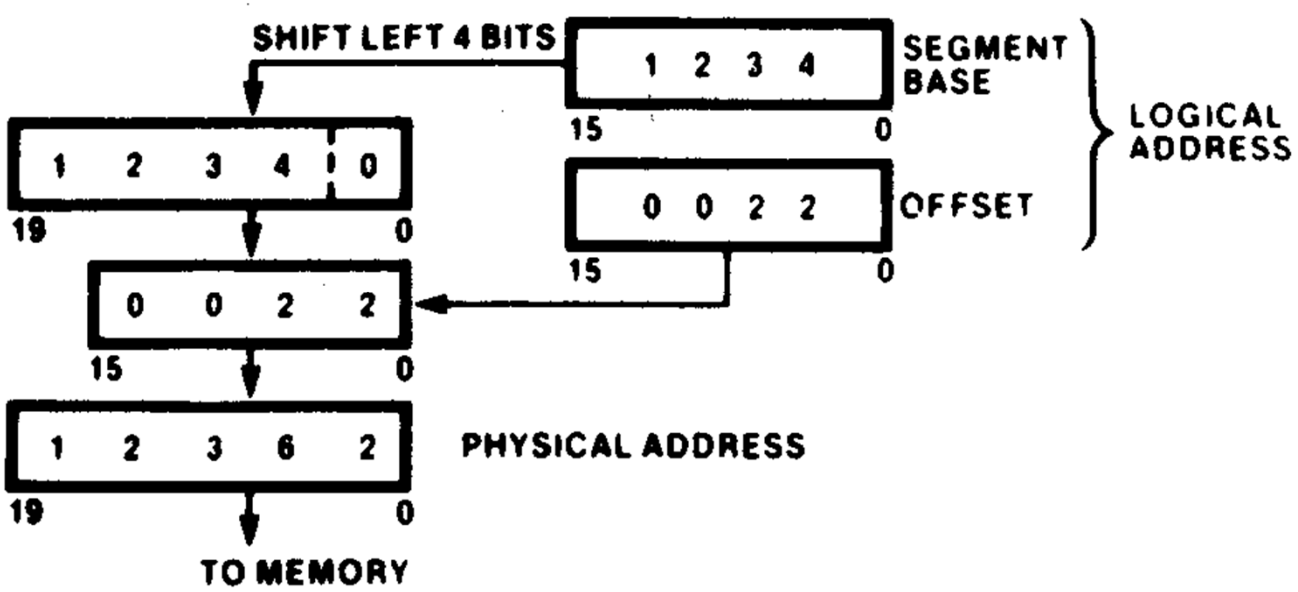
Generating a Memory Address

- Segment base address represents the starting location of the 64 Kbyte segments in memory
  - The lowest address byte in the segment



Generating a Memory Address

- Physical address calculation example
  - Segment base value in a segment register and an offset value are combined to form a physical address



Generating a Memory Address

- Example
  - Segment base value: $1234_{16}=0001001000110100_2$
  - Offset value: $0022_{16}=0000000000100010_2$
  - Shifting left 4 positions and filling with zero results in the segments address
  - $12340_{16}=00010010001101000000_2$
  - $0001001000110100000_2+0000000000100010_2$ $=00010010001101100010_2=12362_{16}=12362H$

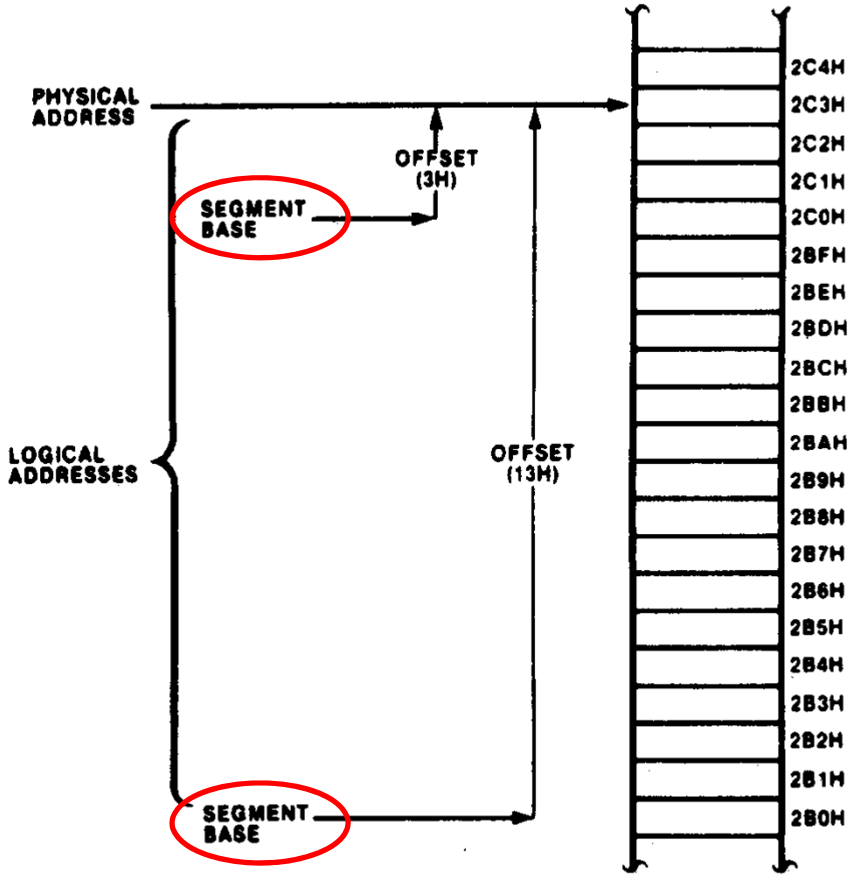
Generating a Memory Address

- Example: What would be the offset required to map to physical address location 002C316 if the contents of the corresponding segment register are $002A_{16}$?

- *Solution:*
  - The offset value can be obtained by shifting the contents of the segment of the segment register left by four bit positions and then subtracting from the physical address. Shifting left give
    - $002A0_{16}$
  - Now subtracting, we get the value of the offset:
    - $002C3_{16} - 002A0_{16} = 0023_{16}$

Generating a Memory Address

- Different logical addresses can be mapped to the same physical address location in memory

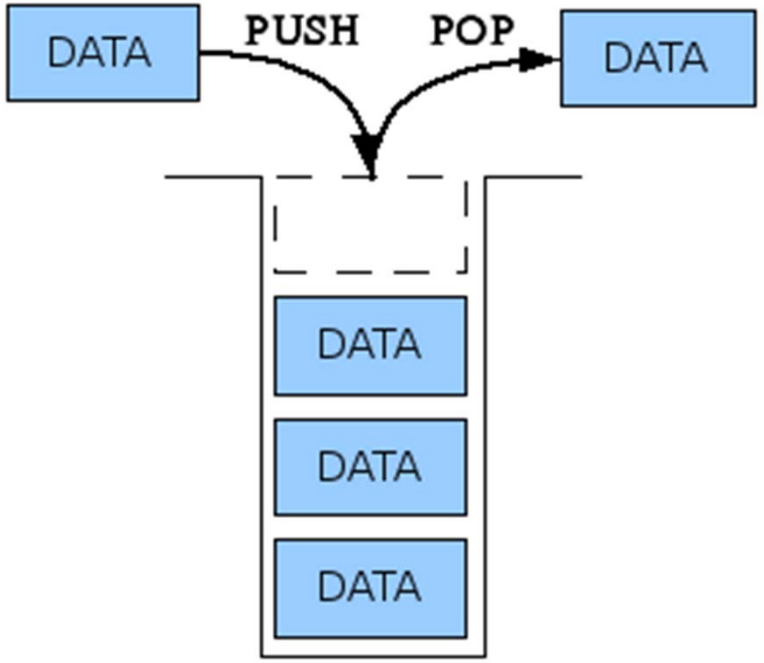


The Stack

- The stack is implemented for temporary storage of information such as data or addresses
- The stack is 64KBytes long and is organized from a software point of view as 32K words
- The contents of the SP and BP registers are used as offsets into the stack segment memory while the segment base value is in the SS register

The Stack

- When the instruction PUSH SI is executed, the contents of SI is pushed onto the stack
- When the instruction POP SI is executed, the top of the stack is popped back into SI

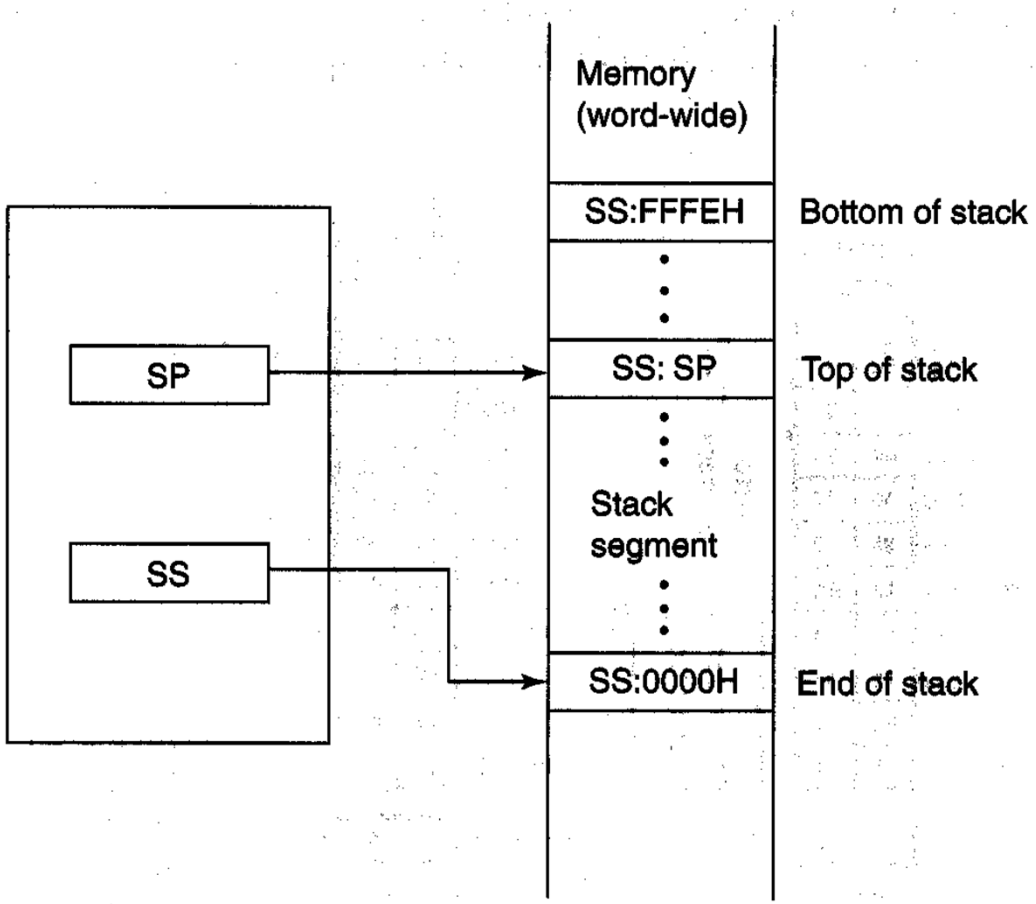


The Stack

- Top of the stack (TOS) SS:SP
  - Physical address of the last storage location in the stack to which data were pushed
  - SP contains an offset value that points to a storage location in the current stack segment (SS:SP)
- Bottom of the stack (BOS) SS:FFFEH
  - At the microcomputer's start-up, SP value is initialized to $FFFE_{16}$
  - Combining this value with the current value in SS gives the highest-addressed word location in the stack

The Stack

- Stack segment of memory



The Stack

- The 8088 can push word-wide data and address information onto the stack from registers or memory
- Only one in many stacks is active at a time

- Example (Push operation)
  - BOS =

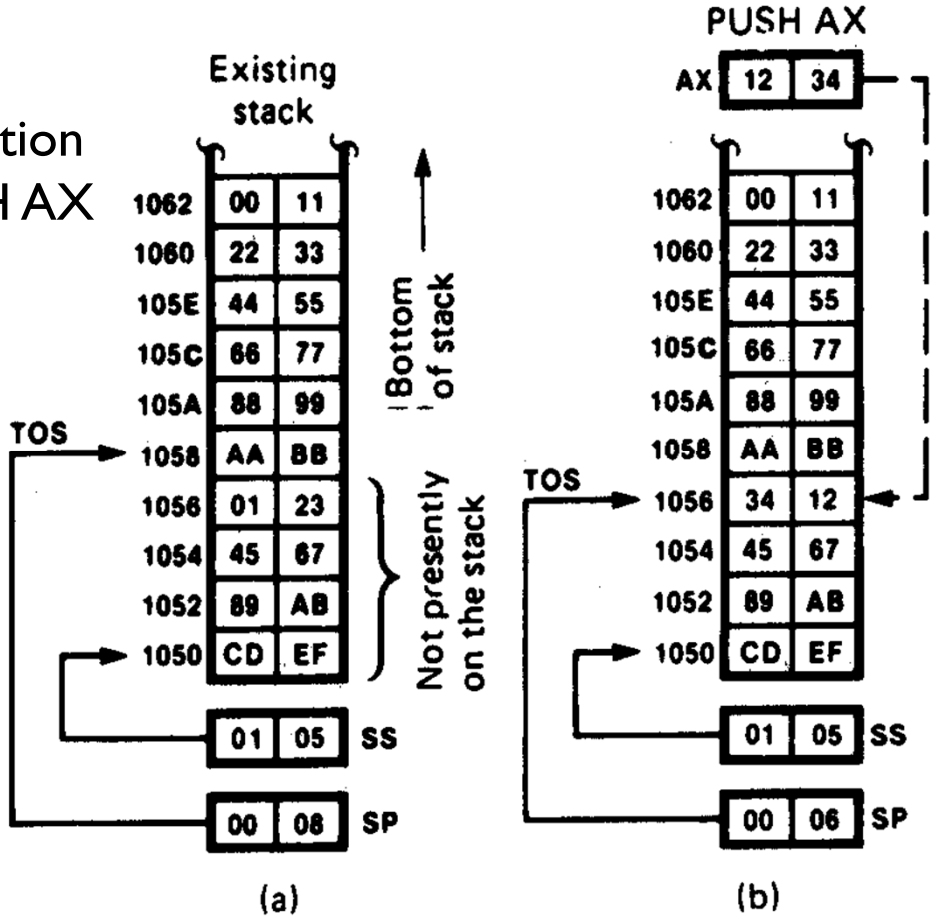
    $01050_{16} + FFFE_{16} = 01104_{16}$
  - TOS =

    $01050_{16} + 0008_{16} = 01058_{16}$

The Stack

- Push operation

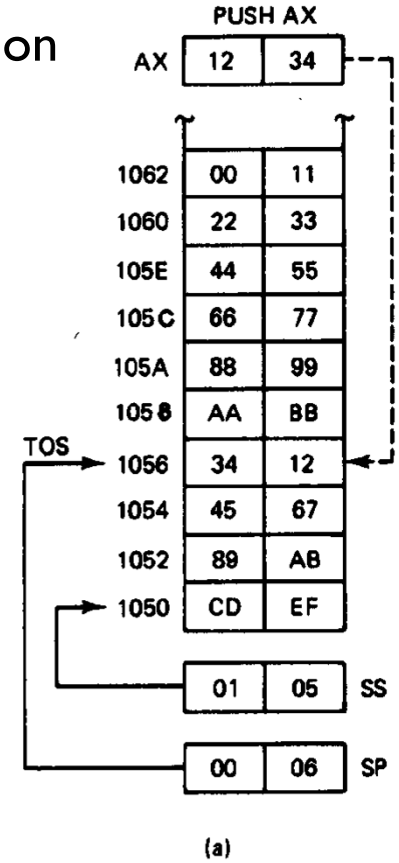
(a) Stack just prior to push operation
(b) Stack after execution of PUSH AX

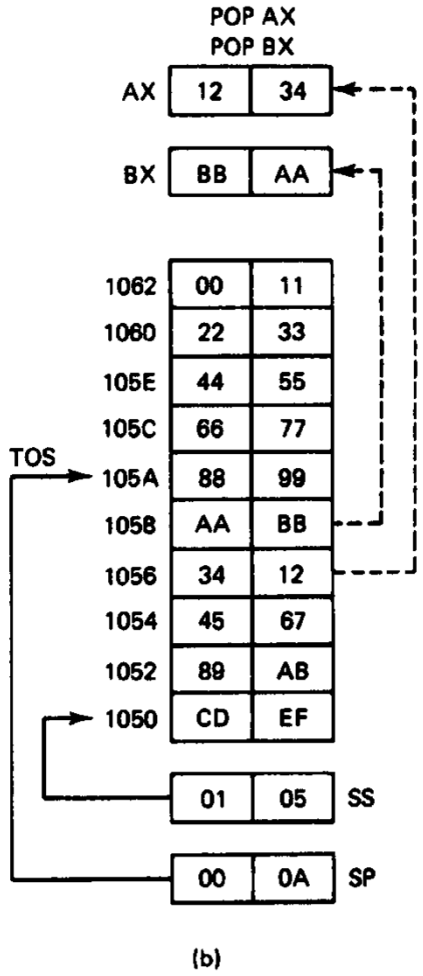


The Stack

- Pop operation

(a) Stack just prior to pop operation
(b) Stack after execution
of POP AX and POP BX





Input/Output Address Space

- 8088 has separate memory and input/output (I/O) address space

- I/O address space is the place where I/O interfaces, such as printer and terminal ports, are implemented
- I/O address range is from 0000_{16} to $FFFF_{16}$
  - This represents 64KByte addresses
- I/O addresses are 16 bits long
  - Each of these addresses corresponds to one byte-wide I/O port

Input/Output Address Space

- I/O address space