1

SOFTWARE ARCHITECTURE OF THE 8088 AND 8086 MICROPROCESSORS

Sarat Kumar Patra
Department of Electronics and Communication Engineering
Rourkela
India-769008

SOFTWARE ARCHITECTURE OF THE 8088 AND 8086 MICROPROCESSORS

» 2.1 Microarchitecture of the 8088/8086 Microprocessor
» 2.2 Software Model of the 8088/8086 Microprocessor
» 2.3 Memory Address Space and Data Organization
» 2.4 Data Types
» 2.5 Segment Registers and Memory Segmentation
» 2.6 Dedicated, Reserved, and General-Used Memory
» 2.7 Instruction Pointer

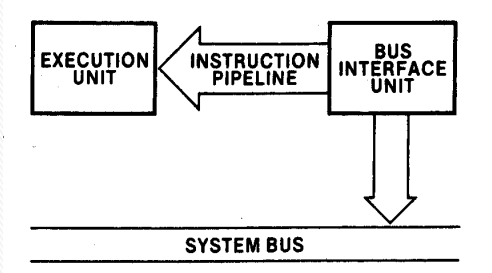
SOFTWARE ARCHITECTURE OF THE 8088 AND 8086 MICROPROCESSORS

» 2.8 Data Registers
» 2.9 Pointer and Index Register
» 2.10 Status Register
» 2.11 Generating a Memory Address
» 2.12 The Stack
» 2.13 Input/ Output Address Space

2.1 Microarchitecture of the 8088/8086 Microprocessor

- 8088/8086 both employ ***parallel processing***
- 8088/8086 contain two processing unit – the ***bus interface unit*** (BIU) and ***execution unit*** (EU)
- The bus interface unit is the path that 8088/8086 connects to external devices.
- The system bus includes an 8-bit bidirectional data bus for 8088 (16 bits for the 8086), a 20-bit address bus, and the signal needed to control transfers over the bus.

2.1 Microarchitecture of the 8088/8086 Microprocessor

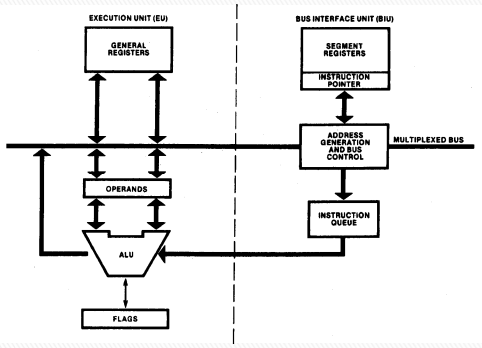


2.1 Microarchitecture of the 088/8086 Microprocessor

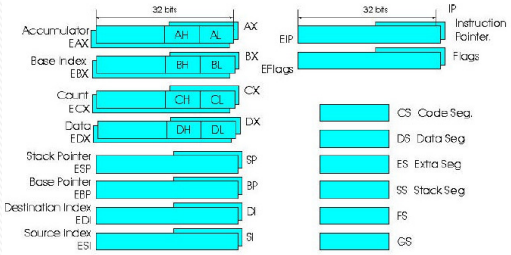
Components in BIU
- Segment register
- The instruction pointer
- Address generation adder
- Bus control logic
- Instruction queue

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

2.1 Microarchitecture of the 8088/8086 Microprocessor

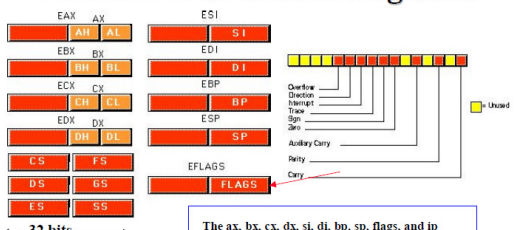


Intel 80x86 Register Organization



32-bit registers not present in 8086, 8088, or 80286

80386/80486/Pentium Registers



The ax, bx, cx, dx, si, di, bp, sp, flags, and ip registers of the 8068 were all extended to 32 bits.

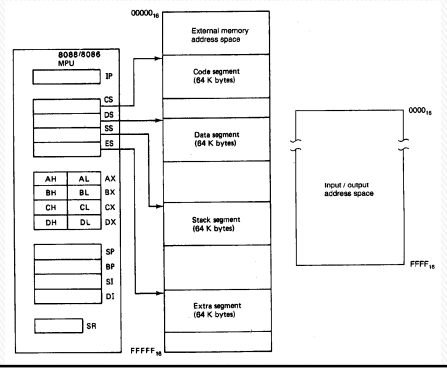
The 80386 calls these new 32 bit versions EAX, EBX, ECX, EDX, ESI, EDI, EBP, ESP, EFLAGS, and EIP

Memory segments can be as large as 4GB

2.2 Software Model of the 8088/8086 Microprocessor

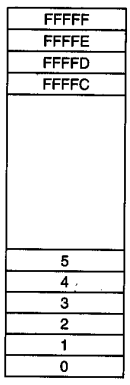
- 8088 microprocessor includes 13 16-bit internal registers.
  - The instruction pointer, IP
  - Four data registers, AX, BX, CX, DX
  - Two pointer registers, BP, SP
  - Two index registers, SI, DI
  - Four segment registers, CS, DS, SS, ES
- The status register, SR, with nine of its bits implemented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

2.2 Software Model of the 8088/8086 Microprocessor



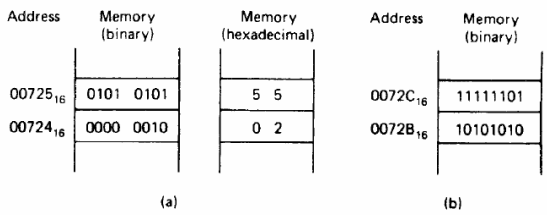
2.3 Memory Address Space and Data Organization

- 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.



2.3 Memory Address Space and Data Organization

» Lower address byte and higher address byte



The two bytes represent the word $0101010100000010_2 = 5502_{16}$

2.3 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?

2.3 Memory Address Space and Data Organization

» *Solution:*

> $11111101_2 = FD_{16} = FDH$

> $10101010_2 = AA_{16} = AAH$

» Together the two bytes give the word

> $1111110110101010_2 = FDAA_{16} = FDAAH$

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

> $0072BH = 0072B_{16} = 00000000011100101011_2$

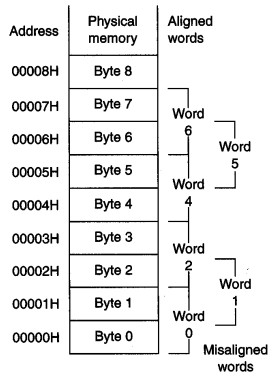
» Therefore, it is misaligned word of data.

2.3 Memory Address Space and Data Organization

» Even- or odd-addressed word

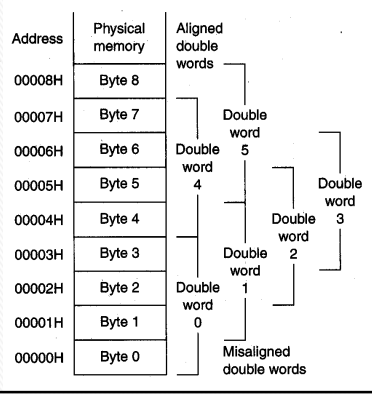
> If the least significant bit of the address is 0, the word is said to be held at an even-addressed boundary.

» Aligned word or misaligned word



2.3 Memory Address Space and Data Organization

» A ***double word*** corresponds to four consecutive bytes of data stored in memory.

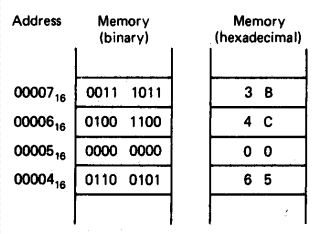


2.3 Memory Address Space and Data Organization

» A pointer is a double word.

> The higher address word represents the ***segment base address***

> The lower address word represents the ***offset***.

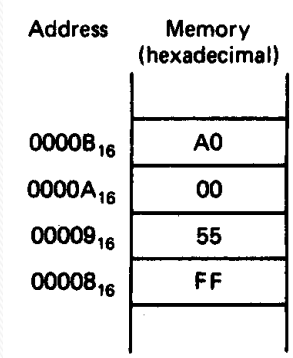


○ *Example:*

- Segment base address = $3B4C_{16} = 0011101101001100_2$
- Offset value = $0065_{16} = 0000000001100101_2$

2.3 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?

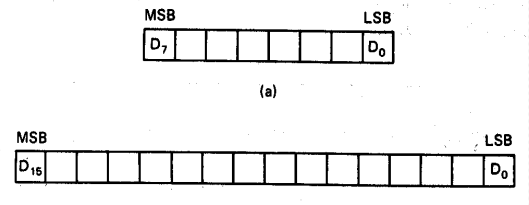


2.3 Memory Address Space and Data Organization

- *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} in the previous figure.
  - 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.

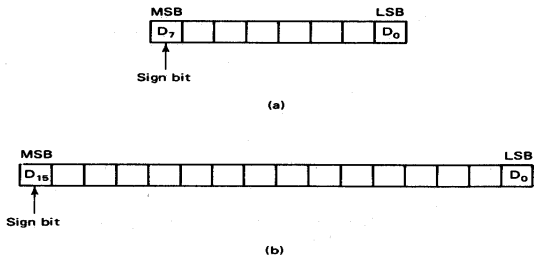
2.4 Data Types

- 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.



2.4 Data Types

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



2.4 Data Types

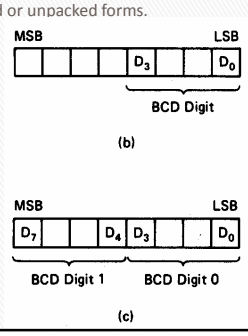
- *EXAMPLE:*
  - A signed word integer equals FEFF16. What decimal number does it represent?
- *Solution:*
  - $FEFF_{16} = 1111111011111111_2$
  - The most significant bit is 1, the number is negative and is in 2's complement form.
  - Converting to its binary equivalent by subtracting 1 from the least significant bit
  - Then complement all bits give:
    - $FEFF_{16} = -0000000100000001_2 = -257$

2.4 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)

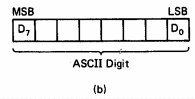


2.4 Data Types

- *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
  - $10010001_2 = 1001_{BCD}0001_{BCD} = 91_{10}$

2.4 Data Types

- The ASCII (American Standard Code for Information Interchange) digit



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)

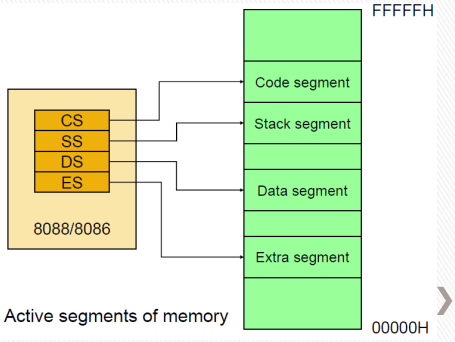
2.4 Data Types

- *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

2.5 Segment Registers and Memory Segmentation

- A ***segment*** represents an independently addressable unit of memory consisting of 64K consecutive bytewide storage locations.
- Each segment is assigned a ***base address*** that identifies its starting point.
- Only four segments can be active at a time:
  - The code segment
  - The stack segment
  - The data segment
  - The extra segment
- The addresses of the active segments are stored in the four internal segment registers: CS, SS, DS, ES.

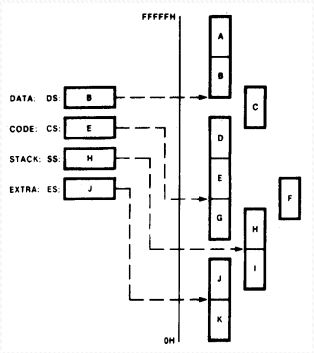
2.5 Segment Registers and Memory Segmentation



2.5 Segment Registers and Memory Segmentation

- Four segments give a maximum of 256Kbytes of active memory.
  - Code segment – 64K
  - Stack – 64K
  - Data storage – 128K
- The base address of a segment must reside on a 16-byte address boundary.
- User accessible segments can be set up to be contiguous, adjacent, disjointed, or even overlapping.

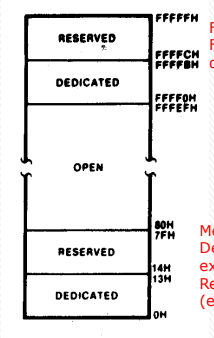
2.5 Segment Registers and Memory Segmentation



2.6 Dedicated, Reserved, and General-Used Memory

- The ***dedicated memory*** (00000_{16} ~ 00013_{16}) are used for storage of the pointers to 8088's internal interrupt service routines and exceptions.
- The ***reserved memory*** (00014_{16} ~ $0007F_{16}$) are used for storage of the pointers to user-defined interrupts.
- The 128-byte dedicated and reserved memory can contain 32 interrupt pointers.
- The ***general-use memory*** (00080_{16} ~ $FFFEF_{16}$) stores data or instructions of the program.
- The dedicated memory ($FFFE0_{16}$ ~ $FFFEB_{16}$) are used for hardware reset jump instruction.

2.6 Dedicated, Reserved, and General-Used Memory



FFFFCH – FFFFFH: Future processor
FFFF0H – FFFFBH: Used for dedicated function (system)

Memory reserved for ISR
Dedicated for internal interrupt and exception
Reserved for user defined interrupt (each interrupt need 4bytes)

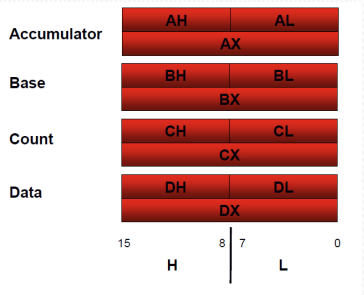
2.7 Instruction Pointer

- The ***instruction pointer (IP)*** identifies the location of the next word of instruction code to be fetched from the current code segment of memory.
- The offset in IP is combined with the current value in 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.

2.8 Data Registers

- Data registers are used for temporary storage of frequently used intermediate results.
- The contents of the data registers can be read, loaded, or modified through software.
- The four data registers are:
  - > Accumulator register, A
  - > Base register, B
  - > Counter register, C
  - > Data register, D
- Each register can be accessed either as a whole (16 bits) for word data or as 8-bit data for byte-wide operation.

2.8 Data Registers



General-purpose data registers of 8088 microprocessor

2.8 Data Registers

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

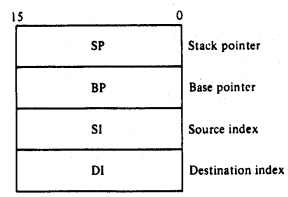
Dedicated register functions

2.9 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 the data segment or extra segment.
- The pointer registers are used to store offset addresses of memory location relative to the stack segment 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.

2.9 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.

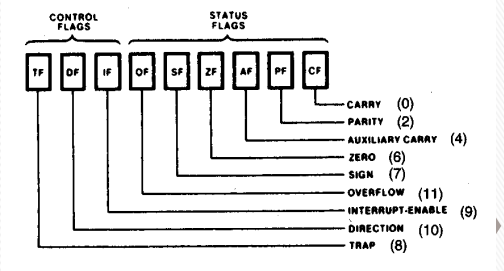


2.10 Status Register

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

2.10 Status Register

- Status and control bits maintained in the flags register
- Generally Set and Tested Individually
- 9 1-bit flags in 8086; 7 are unused



2.10 Status Register

- Status flags indicate current processor status.
  - > **CF** Carry Flag — Arithmetic Carry/Borrow
  - > **OF** Overflow Flag — Arithmetic Overflow
  - > **ZF** Zero Flag — Zero Result; Equal Compare
  - > **SF** Sign Flag — Negative Result; Non-Equal Compare
  - > **PF** Parity Flag — Even Number of "1" bits
  - > **AF** Auxiliary Carry — Used with BCD Arithmetic

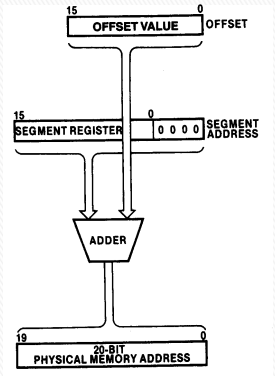
2.10 Status Register

- Control flags influence the 8086 during execution phase
  - > **DF** Direction Flag Auto Increment/Decrement
    - + used for "string operations"
  - > **IF** Interrupt Flag Enables Interrupts
    - + allows "fetch-execute" to be interrupted
  - > **TF** Trap Flag Allows Single-Step
    - + for debugging; causes interrupt after each op

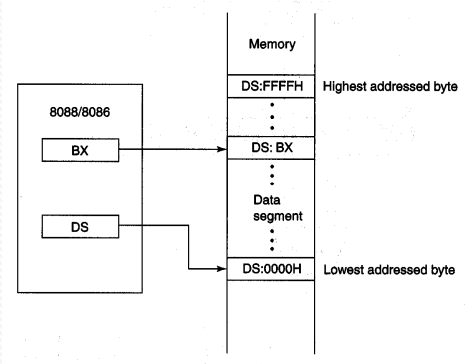
2.11 Generating a Memory Address

- A ***logical address*** in the 8088 microcomputer system is described by a segment base and an offset.
- The ***physical addresses*** that are used to access memory are 20 bits in length.
- The generation of the physical address involves 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.

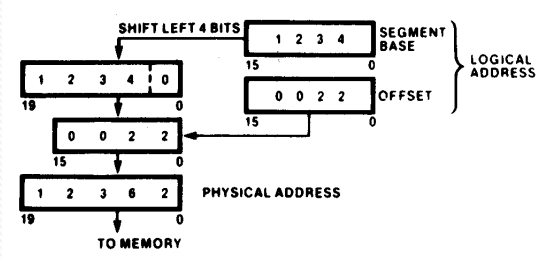
2.11 Generating a Memory Address



2.11 Generating a Memory Address



2.11 Generating a Memory Address

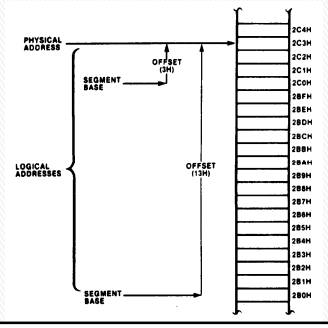


2.11 Generating a Memory Address

- *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:
    - + $002C3_{16} - 002A0_{16} = 0023_{16}$

2.11 Generating a Memory Address

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



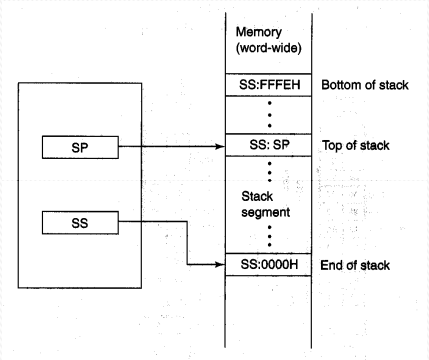
2.12 The Stack

» 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.

2.12 The Stack

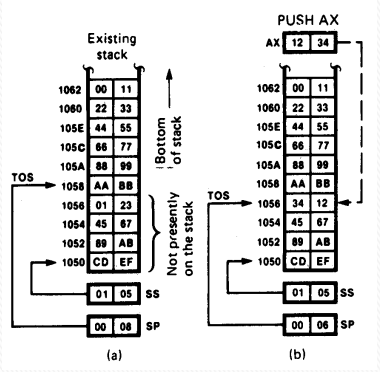
» Push instructions (PUSH) and pop instructions (POP)
» ***Top of the stack*** (TOS) and ***bottom of the stack*** (BOS)
» The 8088 can push **word-wide data** and address information onto the stack from registers or memory.
» Many stacks can exist but only one is active at a time.

2.12 The Stack



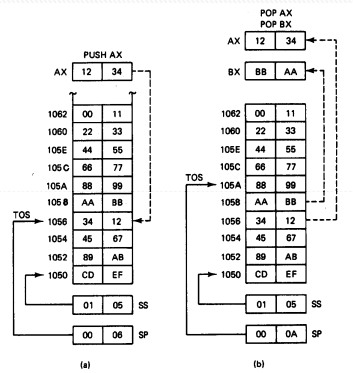
2.12 The Stack

» *EXAMPLE:*
> Push operation
> ABOS = $01050_{16} + FFFE_{16} = 01104_{16}$
> ATOS = $01050_{16} + 0008_{16} = 01058_{16}$



2.12 The Stack

» EXAMPLE
> Pop operation



2.13 Input/Output Address Space

- The 8088 has separate memory and input/output (I/O) address space.
- The I/O address space is the place where I/O interfaces, such as printer and terminal ports, are implemented.
- The I/O address range is from 000016 to FFFF16.
  - > This represents 64KByte addresses.
- The I/O addresses are 16 bits long
  - > Each of these addresses corresponds to one byte-wide I/O port.
- Certain I/O instructions can only perform operations to addresses 000016 thru 00FF16 (***page 0***).

2.13 Input/Output Address Space