Department of Computer Engineering Techniques (Stage: 4)

Advance Computer Technologies

Dr.: Hussein Ali Ameen

hussein_awadh@mustaqbal-college.edu.iq

MORE ABOUT SEGMENTS IN THE 80x86

What is a stack, and why is it needed?

The *stack* is a section of read/write memory (RAM) used by the CPU to store information temporarily. The CPU needs this storage area since there are only a limited number of registers.

The main disadvantage of the stack is its access time. Since the stack is in RAM, it takes much longer to access compared to the access time of registers. After all, the registers are inside the CPU and RAM is outside. This is the reason that some very powerful (and consequently, expensive) computers do not have a stack; the CPU has a large number of registers to work with.

Dr.: Hussein Ali Ameen
hussein_awadh@mustaqbal-college.edu.iq

How stacks are accessed

The two main registers used to access the stack are the SS (stack segment) register and thc SP (stack pointer) register.

When an instruction pushes or pops a general-purpose register, it must be the entire 16-bit register. In other words, one must code "PUSH AX"; there are no instructions such as "PUSH AL" or "PUSH AH".

Dr.: Hussein Ali Ameen
hussein_awadh@mustaqbal-college.edu.iq

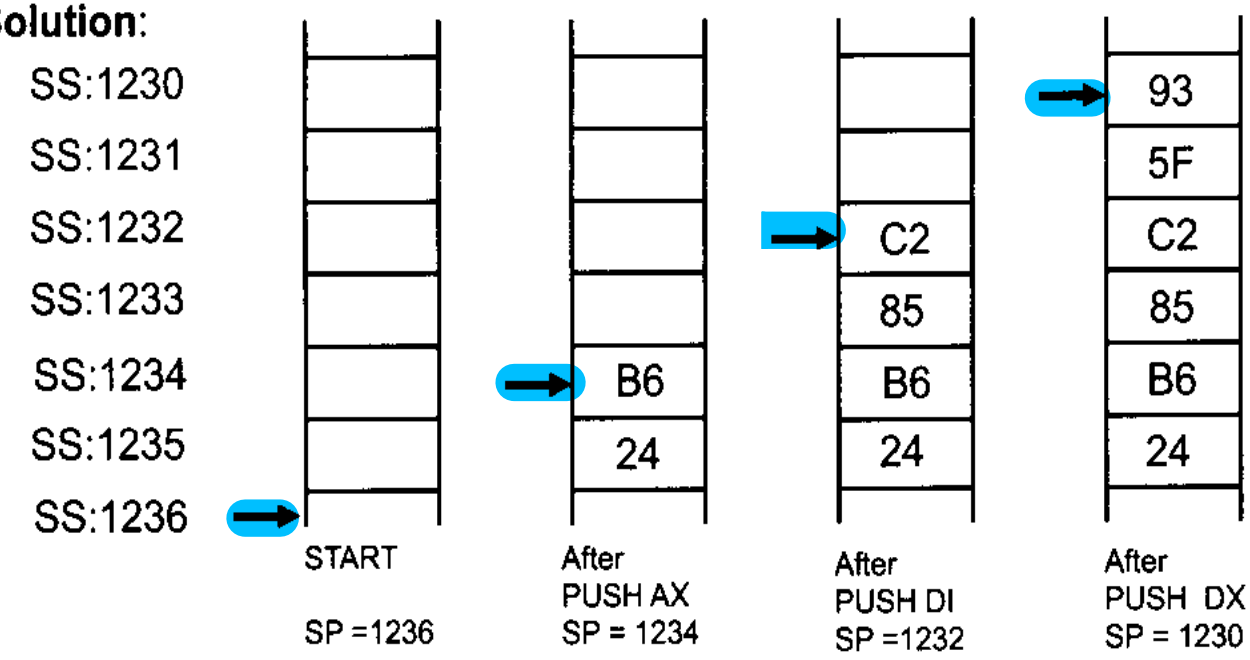
Pushing onto the stack

Example 1-6

Assuming that SP = 1236, AX = 24B6, DI = 85C2, and DX = 5F93, show the contents of the stack as each of the following instructions is executed:

```
PUSH  AX
PUSH  DI
PUSH  DX
```

Solution:



Dr.: Hussein Ali Ameen
hussein_awadh@mustaqbal-college.edu.iq

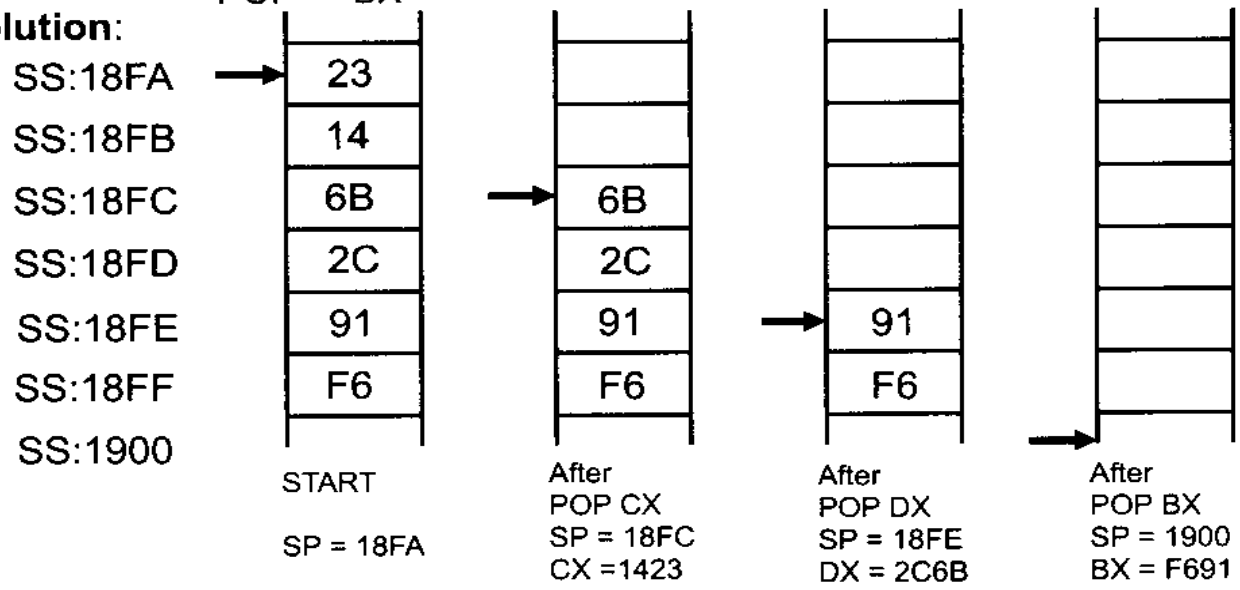
Popping the stack

Example 1-7

Assuming that the stack is as shown below, and SP = 18FA, show the contents of the stack and registers as each of the following instructions is executed:

```
POP   CX
POP   DX
POP   BX
```

Solution:



Dr.: Hussein Ali Ameen
hussein_awadh@mustaqbal-college.edu.iq

Logical address vs. physical address for the stack

What values are assigned to the SP and SS, and who assigns them? It is the job of the DOS operating system to assign the values for the SP and SS since memory management is the responsibility of the operating system. Before leaving the discussion of the stack, two points must be made. First, in the 80x86 literature, the top of the stack is the last stack location occupied. This is different from other CPUs. Second, BP is another register that can be used as an offset into the stack.

Example 1-8

If SS = 3500H and the SP is FFFEH,

(a) Calculate the physical address of the stack.
(b) Calculate the lower range.
(c) Calculate the upper range of the stack segment.
(d) Show the logical address of the stack.

Solution:

(a) 44FFE (35000 + FFFE)
(b) 35000 (35000 + 0000)
(c) 44FFF (35000 + FFFF)
(d) 3500:FFFE

Dr.: Hussein Ali Ameen
hussein_awadh@mustaqbal-college.edu.iq

A few more words about segments in the 80x86

Can a single physical address belong to many different logical addresses?

Logical address (hex)	Physical address (hex)
1000:5020	15020
1500:0020	15020
1502:0000	15020
1400:1020	15020
1302:2000	15020

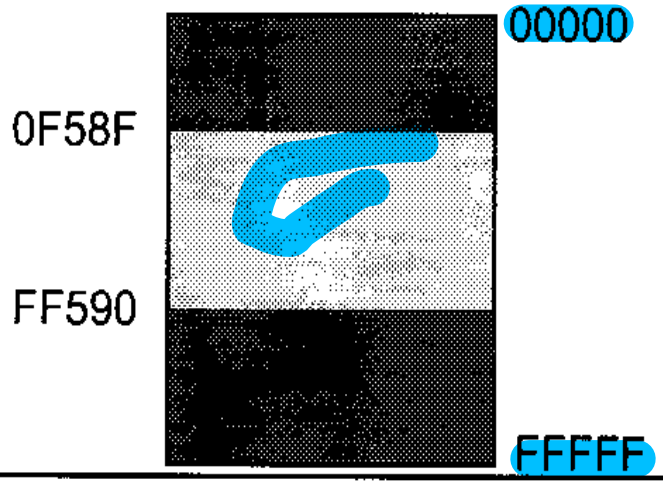
Dr.: Hussein Ali Ameen
hussein_awadh@mustaqbal-college.edu.iq

Example 1-9

What is the range of physical addresses if CS = FF59?

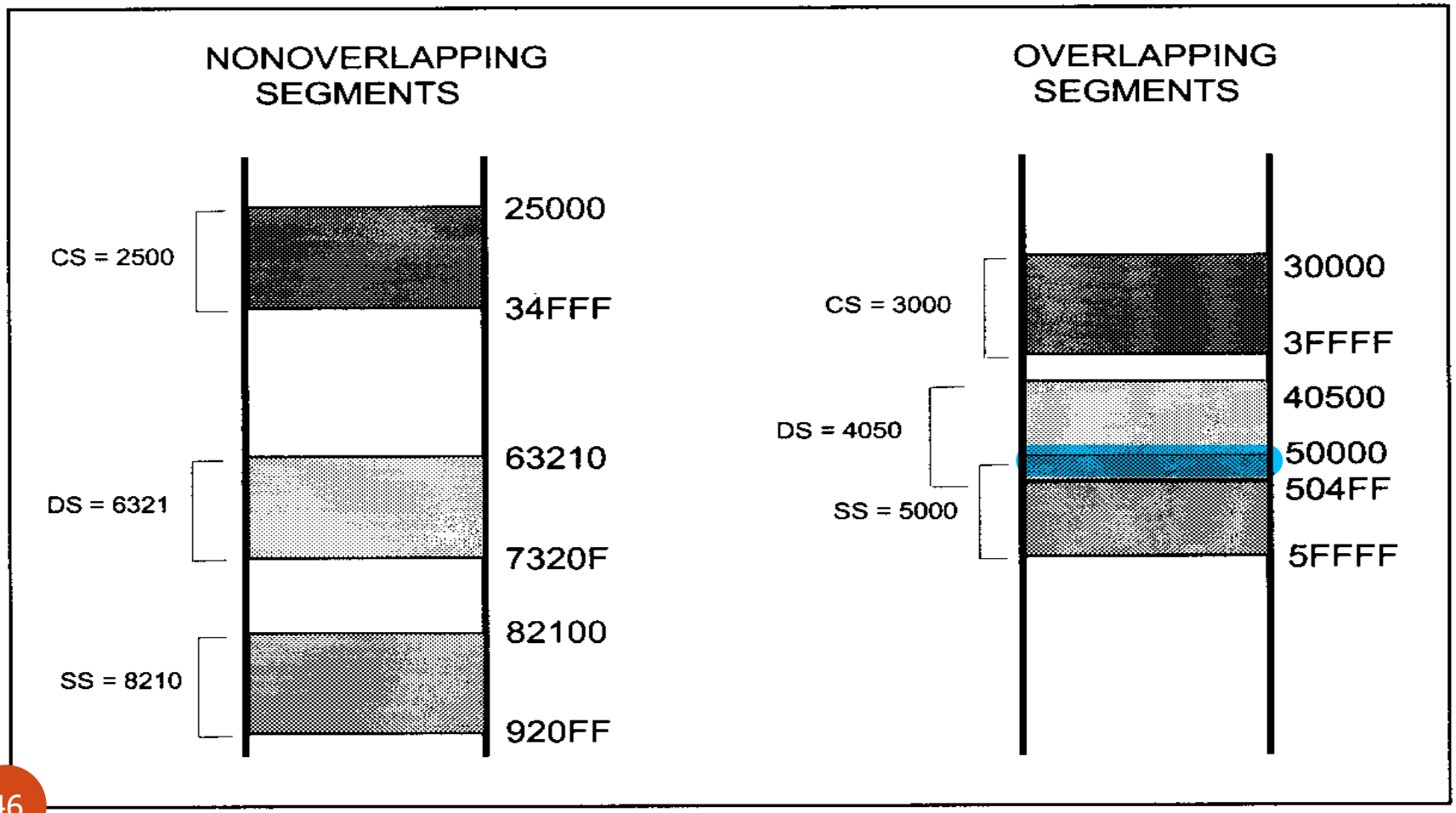
Solution:

The low range is FF590 (FF590 + 0000). The range goes to FFFFF and wraps around, from 00000 to 0F58F (FF590 + FFFF = 0F58F), which is illustrated below.



Dr.: Hussein Ali Ameen
hussein_awadh@mustaqbal-college.edu.iq

Overlapping



igure 1-4. Nonoverlapping vs. Overlapping Segments

Dr.: Hussein Ali Ameen
hussein_awadh@mustaqbal-college.edu.iq

Flag register

Six of the flags are called conditional flags, meaning that they indicate some condition that resulted after an instruction was executed. These six are CF, PF, AF, ZF, SF, and OF. The three remaining flags are sometimes called control flags since they are used to control the operation of instructions before they are executed. A diagram of the flag register is shown in Figure 1-5.

The 16 bits of the flag register:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	R	R	R	OF	DF	IF	TF	SF	ZF	U	AF	U	PF	U	CF

R = reserved
U = undefined
OF = overflow flag
DF = direction flag
IF = interrupt flag
TF = trap flag
SF = sign flag
ZF = zero flag
AF = auxiliary carry flag
PF = parity flag
CF = carry flag

Figure 1-5. Flag Register

CF, the Carry Flag. This flag is set whenever there is a carry out, either from d7 after an 8-bit operation, or from d15 after a 16-bit data operation.

PF, the Parity Flag. After certain operations, the parity of the result's low-order byte is checked. If the byte has an even number of 1s, the parity flag is set to 1; otherwise, it is cleared.

AF, Auxiliary Carry Flag. If there is a carry from d3 to d4 of an operation, this bit is set; otherwise, it is cleared (set equal to zero). This flag is used by the instructions that perform BCD (binary coded decimal) arithmetic.

ZF, the Zero Flag. The zero flag is set to 1 if the result of an arithmetic or logical operation is zero; otherwise, it is cleared.

SF, the Sign Flag. Binary representation of signed numbers uses the most significant bit as the sign bit. After arithmetic or logic operations, the status of this sign bit is copied into the SF, thereby indicating the sign of the result.

TF, the Trap Flag. When this flag is set it allows the program to single-step, meaning to execute one instruction at a time. Single-stepping is used for debugging purposes.

IF, Interrupt Enable Flag. This bit is set or cleared to enable or disable only the external maskable interrupt requests.

DF, the Direction Flag. This bit is used to control the direction of string operations, which are described in Chapter 6.

OF, the Overflow Flag. This flag is set whenever the result of a signed number operation is too large, causing the high-order bit to overflow into the sign bit. In general, the carry flag is used to detect errors in unsigned arithmetic operations. The overflow flag is only used to detect errors in signed arithmetic operations.

...look at Examples 1-10 and 1-11.

Example 1-10

Show how the flag register is affected by the addition of 38H and 2FH.

Solution:

```
MOV   BH,38H      ;BH= 38H
ADD   BH,2FH      ;add 2F to BH, now BH=67H

      38          0011  1000
+     2F          0010  1111
      67          0110  0111
```

CF = 0 since there is no carry beyond d7
PF = 0 since there is an odd number of 1s in the result
AF = 1 since there is a carry from d3 to d4
ZF = 0 since the result is not zero
SF = 0 since d7 of the result is zero

Dr.: Hussein Ali Ameen
hussein_awadh@mustaqbal-college.edu.iq

Example 1-11

Show how the flag register is affected by

```
MOV   AL,9CH      ;AL=9CH
MOV   DH,64H      ;DH=64H
ADD   AL,DH       ;now AL=0
```

Solution:

```
      9C          1 1001  1100
+     64            0110  0100
      00            0000  0000
```

CF=1 since there is a carry beyond d7
PF=1 since there is an even number of 1s in the result
AF=1 since there is a carry from d3 to d4
ZF=1 since the result is zero
SF=0 since d7 of the result is zero

Example 1-12

Show how the flag register is affected by

```
MOV   AX,34F5H     ;AX= 34F5H
ADD   AX,95EBH     ;now AX= CAE0H
```

Solution:

```
        34F5    0011  0100  1111  0101
+       95EB    1001  0101  1110  1011
        CAE0    1100  1010  1110  0000
```

CF = 0 since there is no carry beyond d15

PF = 0 since there is an odd number of 1s in the lower byte

AF = 1 since there is a carry from d3 to d4

ZF = 0 since the result is not zero

SF = 1 since d15 of the result is one

Dr.: Hussein Ali Ameen
hussein_awadh@mustaqbal-college.edu.iq

Example 1-13

Show how the flag register is affected by

```
MOV   BX,AAAAH     ;BX= AAAAH
ADD   BX,5556H     ;now BX= 0000H
```

Solution:

	AAAA	1010	1010	1010	1010
+	5556	0101	0101	0101	0110
	0000	0000	0000	0000	0000

CF = 1 since there is a carry beyond d15

PF = 1 since there is an even number of 1s in the lower byte

AF = 1 since there is a carry from d3 to d4

ZF = 1 since the result is zero

SF = 0 since d15 of the result is zero

Dr.: Hussein Ali Ameen
hussein_awadh@mustaqbal-college.edu.iq

Example 1-14

Show how the flag register is affected by

```
MOV   AX,94C2H     ;AX=94C2H
MOV   BX,323EH     ;BX=323EH
ADD   AX,BX        ;now AX=C700H
MOV   DX,AX        ;now DX=C700H
MOV   CX,DX        ;now CX=C700H
```

Solution:

```
      94C2      1001  0100  1100  0010
+     323E      0011  0010  0011  1110
      C700      1100  0111  0000  0000
```

After the ADD operation, the following are the flag bits:

CF = 0 since there is no carry beyond d15

PF = 1 since there is an even number of 1s in the lower byte

AF = 1 since there is a carry from d3 to d4

ZF = 0 since the result is not zero

SF = 1 since d15 of the result is 1

Dr.: Hussein Ali Ameen
hussein_awadh@mustaqbal-college.edu.iq

Use of the zero flag for looping

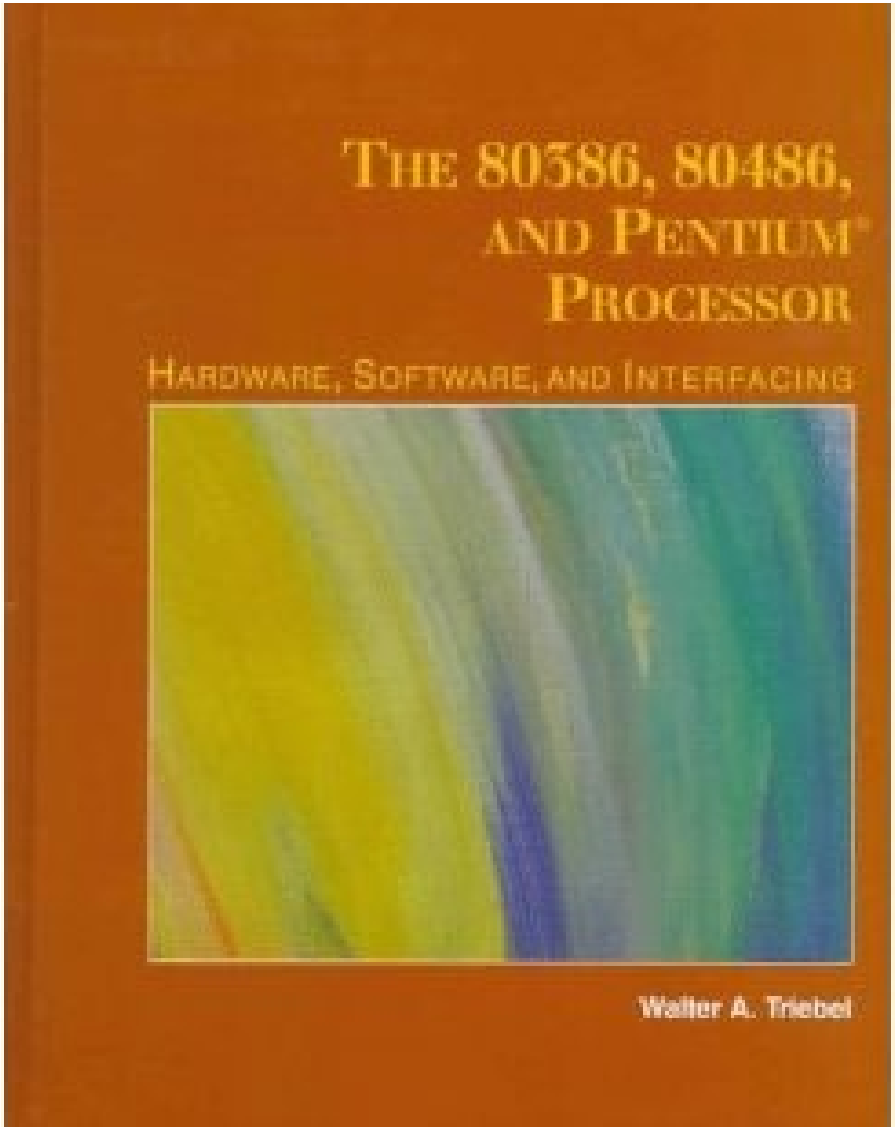
```
        MOV   CX,05       ;CX holds the loop count
        MOV   BX,0200H    ;BX holds the offset data address
        MOV   AL,00       ;initialize AL
ADD_LP: ADD   AL,[BX]     ;add the next byte to AL
        INC   BX          ;increment the data pointer
        DEC   CX          ;decrement the loop counter
        JNZ   ADD_LP      ;jump to next iteration if counter not zero
```

Dr.: Hussein Ali Ameen
hussein_awadh@mustaqbal-college.edu.iq

Review Questions

1. Which registers are used to access the stack?
2. With each PUSH instruction, the stack pointer register SP is (circle one) incremented/decremented by 2.
3. With each POP instruction, SP is (circle one) incremented/decremented by 2.
4. List three possible logical addresses corresponding to physical address 143F0.
5. The ADD instruction can affect which bits of the flag register?
6. The carry flag will be set to 1 in an 8-bit ADD if there is a carry out from bit ___.
7. CF will be set to 1 in a 16-bit ADD if there is a carry out from bit ____.

Text book 1


THE 80386, 80486,
AND PENTIUM®
PROCESSOR
HARDWARE, SOFTWARE, AND INTERFACING
Walter A. Triebel

Text book 2


INTERNATIONAL EDITION
THE 80x86 IBM PC AND COMPATIBLE COMPUTERS (VOLUMES I & II)
ASSEMBLY LANGUAGE, DESIGN, AND INTERFACING
4th Edition
Muhammad Ali Mazidi
Janice Gillispie Mazidi