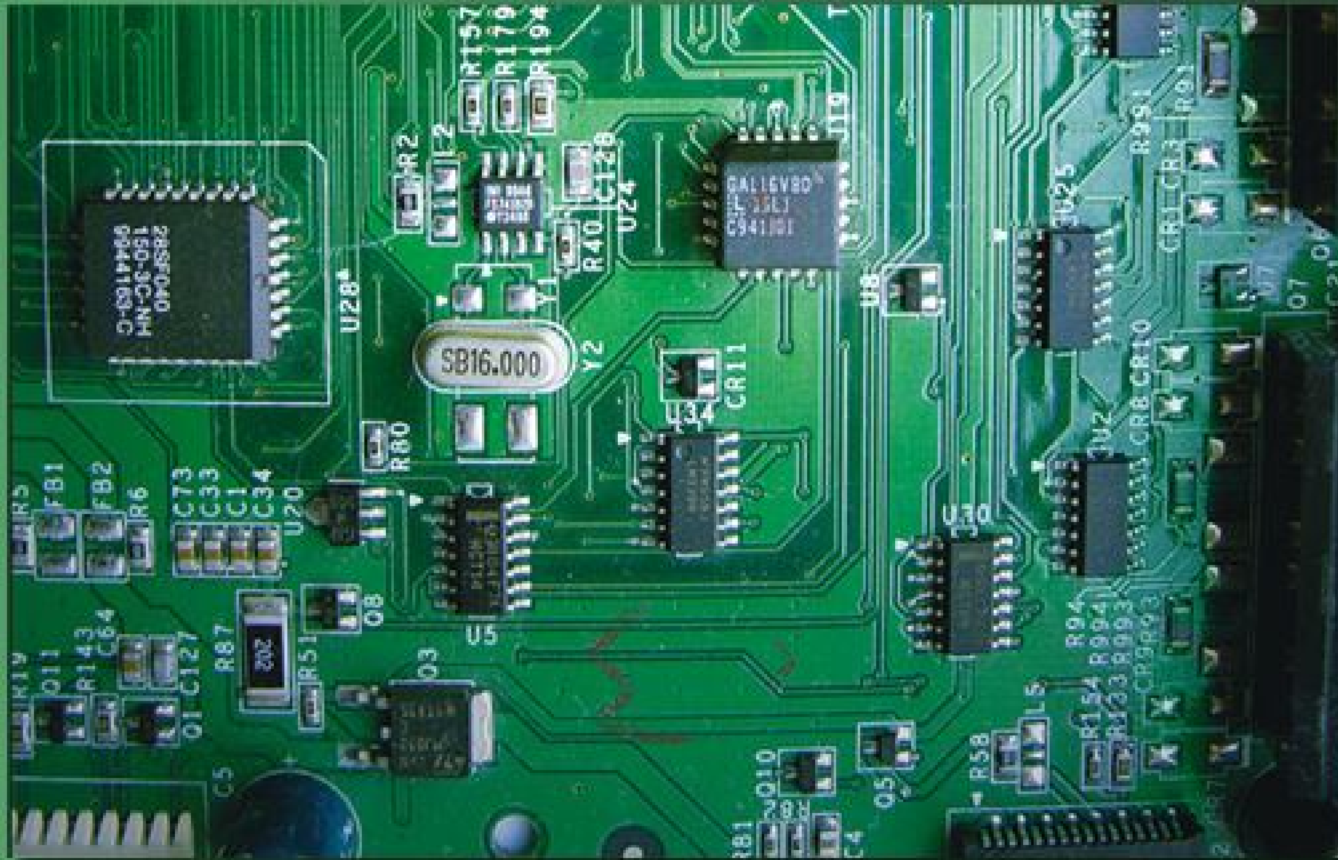
The Intel Microprocessors

8086/8088, 80186/80188, 80286, 80386, 80486 Pentium, Pentium Pro Processor, Pentium II, Pentium 4, and Core2 with 64-bit Extensions

Architecture, Programming, and Interfacing



EIGHTH EDITION

Barry B. Brey

PEARSON

Chapter 6: Program Control Instructions

6–1 THE JUMP GROUP

- Allows programmer to skip program sections and branch to any part of memory for the next instruction.
- A conditional jump instruction allows decisions based upon numerical tests.
  - results are held in the flag bits, then tested by conditional jump instructions
- LOOP and conditional LOOP are also forms of the jump instruction.

Copyright ©2009 by Pearson Education, Inc.
Upper Saddle River, New Jersey 07458 • All rights reserved.

Unconditional Jump (JMP)

- Three types: short jump, near jump, far jump.
- **Short jump** is a 2-byte instruction that allows jumps or branches to memory locations within +127 and –128 bytes.
  - from the address following the jump
- 3-byte **near jump** allows a branch or jump within ±32K bytes from the instruction in the current code segment.

Copyright ©2009 by Pearson Education, Inc.
Upper Saddle River, New Jersey 07458 • All rights reserved.

- 5-byte **far jump** allows a jump to any memory location within the real memory system.
- The short and near jumps are often called **intrasegment jumps.**
- Far jumps are called **intersegment jumps**.

PEARSON

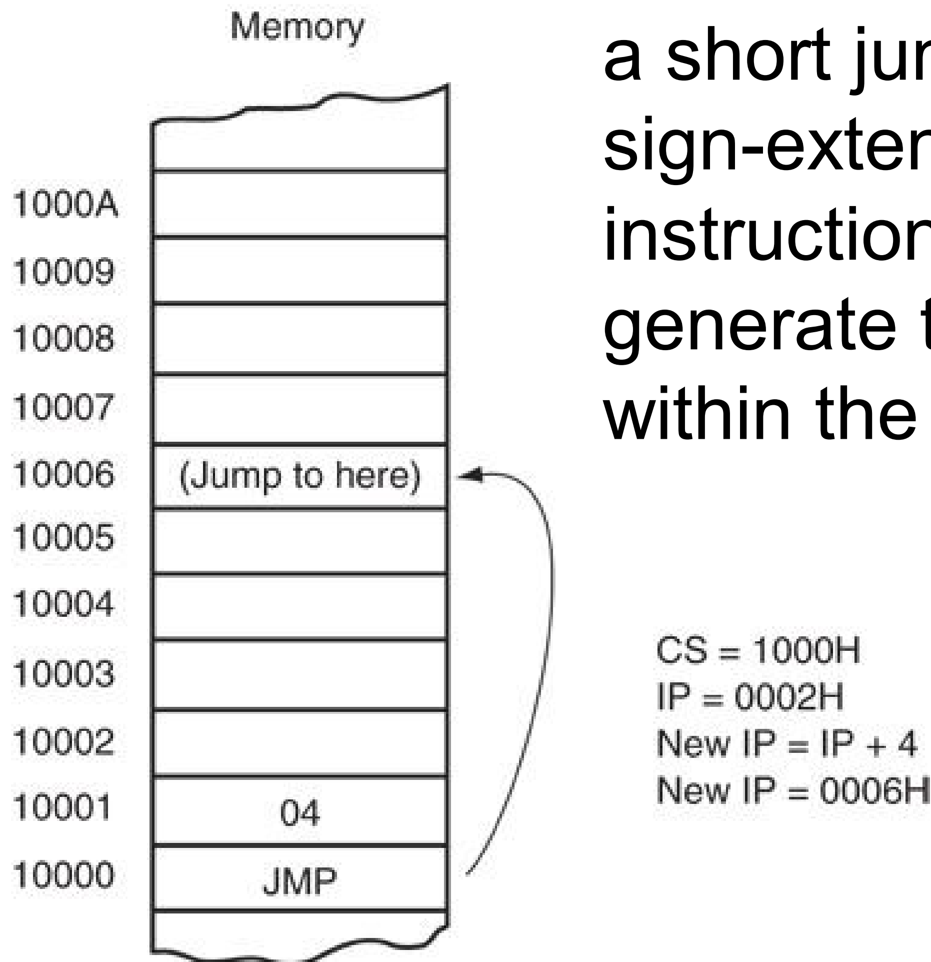
Copyright ©2009 by Pearson Education, Inc.
Upper Saddle River, New Jersey 07458 • All rights reserved.

Short Jump

- Called **relative jumps** because they can be moved, with related software, to any location in the current code segment without a change.
  - jump address is not stored with the opcode
  - a **distance**, or displacement, follows the opcode
- The short jump displacement is a distance represented by a 1-byte signed number whose value ranges between +127 and –128.
- Short jump instruction appears in Figure 6–2.

Copyright ©2009 by Pearson Education, Inc.
Upper Saddle River, New Jersey 07458 • All rights reserved.

Figure 6–2 A short jump to four memory locations beyond the address of the next instruction.



- when the microprocessor executes a short jump, the displacement is sign-extended and added to the instruction pointer (IP/EIP) to generate the jump address within the current code segment

- The instruction branches to this new address for the next instruction in the program

PEARSON

Copyright ©2009 by Pearson Education, Inc.
Upper Saddle River, New Jersey 07458 • All rights reserved.

- When a jump references an address, a label normally identifies the address.
- The JMP NEXT instruction is an example.
  - it jumps to label NEXT for the next instruction
  - very rare to use an actual hexadecimal address with any jump instruction
- The label NEXT must be followed by a colon (NEXT:) to allow an instruction to reference it
  - if a colon does not follow, you cannot jump to it
- The only time a colon is used is when the label is used with a jump or call instruction.

Copyright ©2009 by Pearson Education, Inc.
Upper Saddle River, New Jersey 07458 • All rights reserved.

Example

```
0000  33 db                  xor bx, bx
0002  b8 0001      start:    mov ax, 1
0005  03 c3                  add ax, bx
0007  Eb 17                  jmp short next
```

The Short directive force a short jump

<skipped memory locations>

```
0020  8b d8        next:     mov bx,ax
0022  eb de                  jmp start
```

Assembles as short also; most assemblers choose the best form of the JUMP instruction

Copyright ©2009 by Pearson Education, Inc.
Upper Saddle River, New Jersey 07458 • All rights reserved.

Near Jump

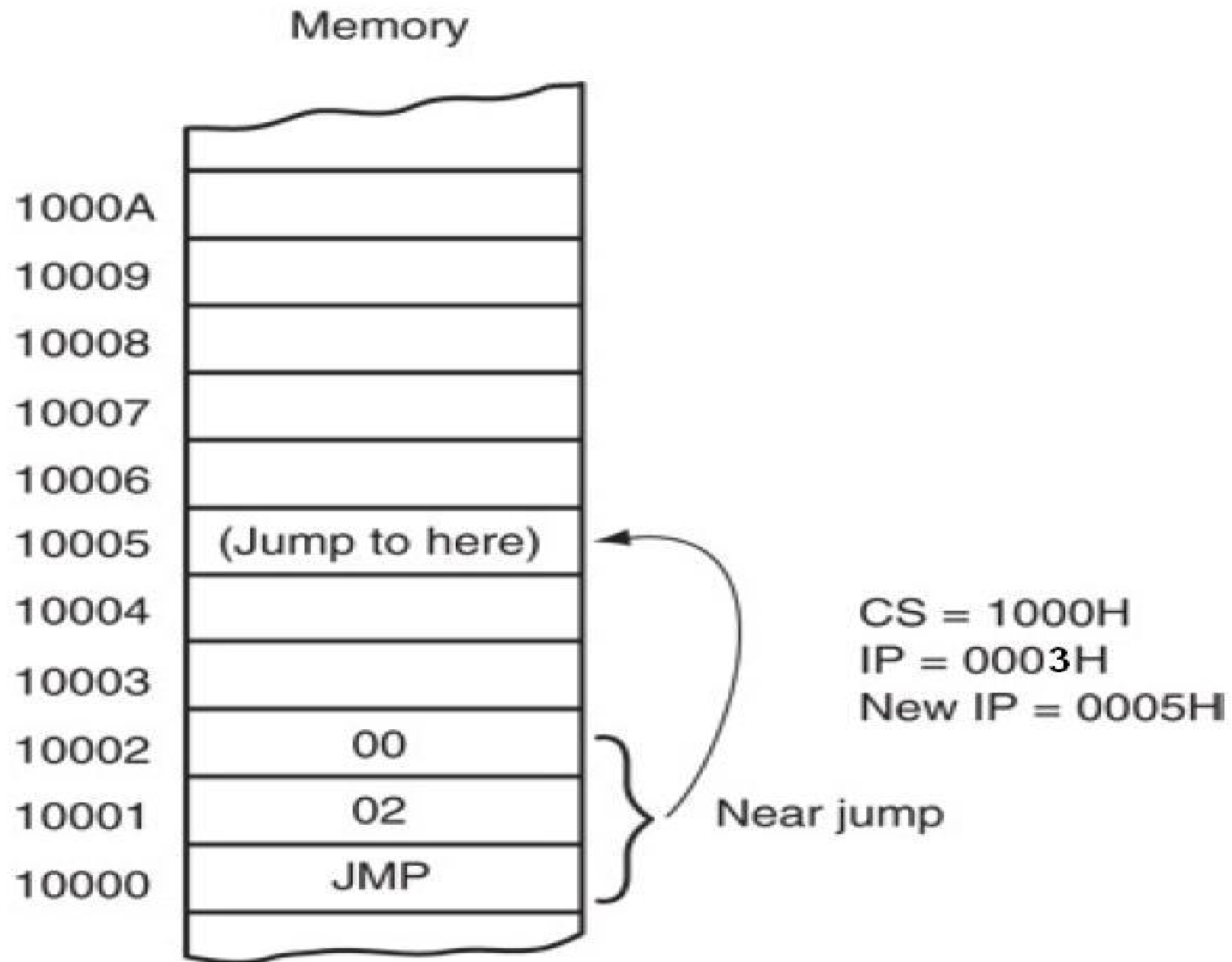
- A near jump passes control to an instruction in the current code segment located within ±32K bytes from the near jump instruction.
  - distance is ±2G in 80386 and above when operated in protected mode
- Near jump is a 3-byte instruction with opcode followed by a signed 16-bit displacement.

Copyright ©2009 by Pearson Education, Inc.
Upper Saddle River, New Jersey 07458 • All rights reserved.

- Signed displacement adds to the instruction pointer (IP) to generate the jump address.
  - because signed displacement is ±32K, a near jump can jump to any memory location within the current real mode code segment
- Figure 6–3 illustrates the operation of the real mode near jump instruction.

Copyright ©2009 by Pearson Education, Inc.
Upper Saddle River, New Jersey 07458 • All rights reserved.

Figure 6–3 A near jump that adds the displacement (0002H) to the contents of IP.



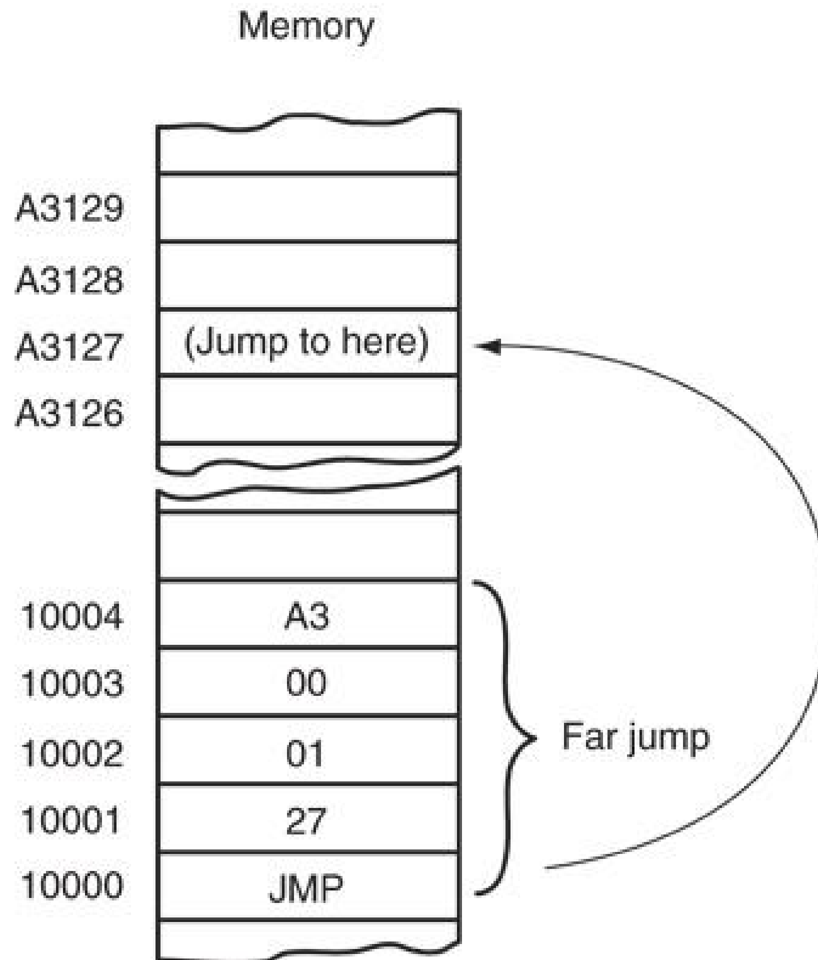
Copyright ©2009 by Pearson Education, Inc.
Upper Saddle River, New Jersey 07458 • All rights reserved.

Far Jump

- Obtains a new segment and offset address to accomplish the jump:
  - bytes 2 and 3 of this 5-byte instruction contain the new offset address
  - bytes 4 and 5 contain the new segment address
  - in protected mode, the segment address accesses a descriptor with the base address of the far jump segment
  - offset address, either 16 or 32 bits, contains the offset address within the new code segment

Copyright ©2009 by Pearson Education, Inc.
Upper Saddle River, New Jersey 07458 • All rights reserved.

Figure 6–4 A far jump instruction replaces the contents of both CS and IP with 4 bytes following the opcode.



IP= 10005

New offset= 0127

- **New CS=A300**
- **New IP=**

Copyright ©2009 by Pearson Education, Inc.
Upper Saddle River, New Jersey 07458 • All rights reserved.

Jumps with Register Operands

- Jump can also use a 16- or 32-bit register as an operand.
  – automatically sets up as an **indirect jump**
  – address of the jump is in the register specified by the jump instruction
- Unlike displacement associated with the near jump, register contents are transferred directly into the instruction pointer.
- An indirect jump does not add to the instruction pointer.

Copyright ©2009 by Pearson Education, Inc.
Upper Saddle River, New Jersey 07458 • All rights reserved.

- JMP AX, for example, copies the contents of the AX register into the IP.
  - allows a jump to any location within the current code segment
- In 80386 and above, JMP EAX also jumps to any location within the current code segment;
  - in protected mode the code segment can be 4G bytes long, so a 32-bit offset address is needed

Copyright ©2009 by Pearson Education, Inc.
Upper Saddle River, New Jersey 07458 • All rights reserved.

Conditional Jumps

- Always short jumps in 8086 - 80286.
  - limits range to within +127 and –128 bytes from the location following the conditional jump
- In 80386 and above, conditional jumps are either short or near jumps (±32K).
  - in 64-bit mode of the Pentium 4, the near jump distance is ±2G for the conditional jumps
- Allows a conditional jump to any location within the current code segment.

Copyright ©2009 by Pearson Education, Inc.
Upper Saddle River, New Jersey 07458 • All rights reserved.

- Conditional jump instructions test flag bits:
  - sign (S), zero (Z), carry (C)
  - parity (P), overflow (0)
- If the condition under test is true, a branch to the label associated with the jump instruction occurs.
  - if false, next sequential step in program executes
  - for example, a JC will jump if the carry bit is set

Copyright ©2009 by Pearson Education, Inc.
Upper Saddle River, New Jersey 07458 • All rights reserved.

- When signed numbers are compared, use the JG, JL, JGE, JLE, JE, and JNE instructions.
- When unsigned numbers are compared, use the JA, JB, JAB, JBE, JE, and JNE instructions.
- Remaining conditional jumps test individual flag bits, such as overflow and parity.

Copyright ©2009 by Pearson Education, Inc.
Upper Saddle River, New Jersey 07458 • All rights reserved.

LOOP

- A combination of a decrement CX and the JNZ conditional jump.
- In 8086 - 80286 LOOP decrements CX.
  - if CX != 0, it jumps to the address indicated by the label
  - If CX becomes 0, the next sequential instruction executes

Copyright ©2009 by Pearson Education, Inc.
Upper Saddle River, New Jersey 07458 • All rights reserved.

Conditional LOOPs

- LOOP instruction also has conditional forms: LOOPE and LOOPNE
- LOOPE (**loop while equal**) instruction jumps if CX != 0 while an equal condition exists.
  - will exit loop if the condition is not equal or the CX register decrements to 0
- LOOPNE (**loop while not equal**) jumps if CX != 0 while a not-equal condition exists.
  - will exit loop if the condition is equal or the CX register decrements to 0

Copyright ©2009 by Pearson Education, Inc.
Upper Saddle River, New Jersey 07458 • All rights reserved.

6–3 PROCEDURES

- A procedure is a group of instructions that usually performs one task.
  – subroutine, method, or **function** is an important part of any system's architecture
- A procedure is a reusable section of the software stored in memory once, used as often as necessary.
  – saves memory space and makes it easier to develop software

Copyright ©2009 by Pearson Education, Inc.
Upper Saddle River, New Jersey 07458 • All rights reserved.

- Disadvantage of procedure is time it takes the computer to link to, and return from it.
  - CALL links to the procedure; the RET (**return**) instruction returns from the procedure
- CALL pushes the address of the instruction following the CALL (**return address**) on the stack.
  - the stack stores the return address when a procedure is called during a program
- RET instruction removes an address from the stack so the program returns to the instruction following the CALL.

Copyright ©2009 by Pearson Education, Inc.
Upper Saddle River, New Jersey 07458 • All rights reserved.

- Procedures that are to be used by all software (global) should be written as far procedures.
- Procedures that are used by a given task (local) are normally defined as near procedures.
- Most procedures are near procedures.

PEARSON
Copyright ©2009 by Pearson Education, Inc.
Upper Saddle River, New Jersey 07458 • All rights reserved.

CALL

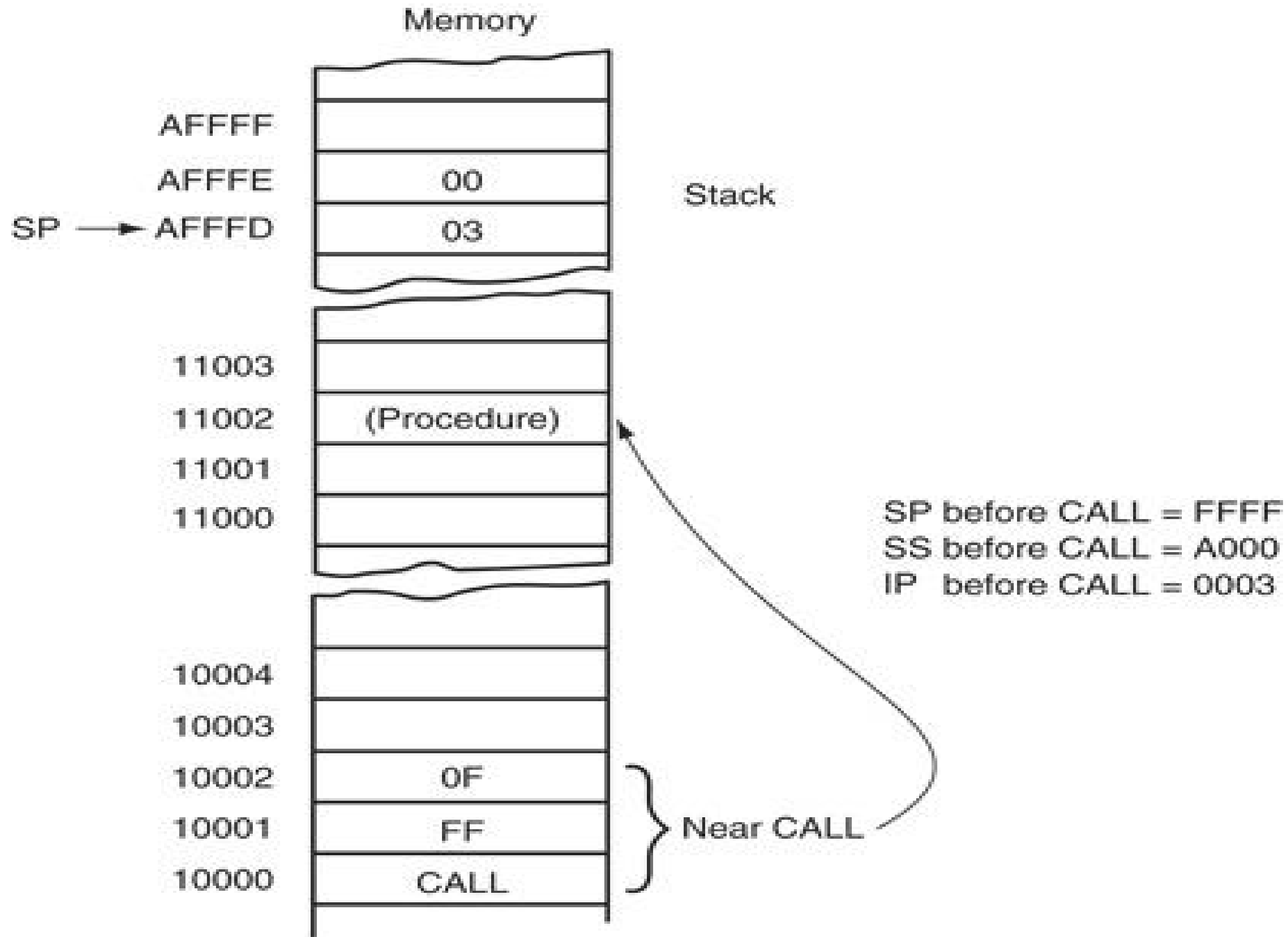
- Transfers the flow of the program to the procedure.
- CALL instruction differs from the jump instruction because a CALL saves a return address on the stack.
- The return address returns control to the instruction that immediately follows the CALL in a program when a RET instruction executes.

Copyright ©2009 by Pearson Education, Inc.
Upper Saddle River, New Jersey 07458 • All rights reserved.

- Why save the IP or EIP on the stack?
  - the instruction pointer always points to the next instruction in the program
- For the CALL instruction, the contents of IP/EIP are pushed onto the stack.
  - program control passes to the instruction following the CALL after a procedure ends
- Figure 6–6 shows the return address (IP) stored on the stack and the call to the procedure.

Copyright ©2009 by Pearson Education, Inc.
Upper Saddle River, New Jersey 07458 • All rights reserved.

Figure 6–6 The effect of a CALL on the stack and the instruction pointer.



Copyright ©2009 by Pearson Education, Inc.
Upper Saddle River, New Jersey 07458 • All rights reserved.

CALLs with Register Operands

- An example CALL BX, which pushes the contents of IP onto the stack.
  - then jumps to the offset address, located in register BX, in the current code segment
- Always uses a 16-bit offset address, stored in any 16-bit register except segment registers.

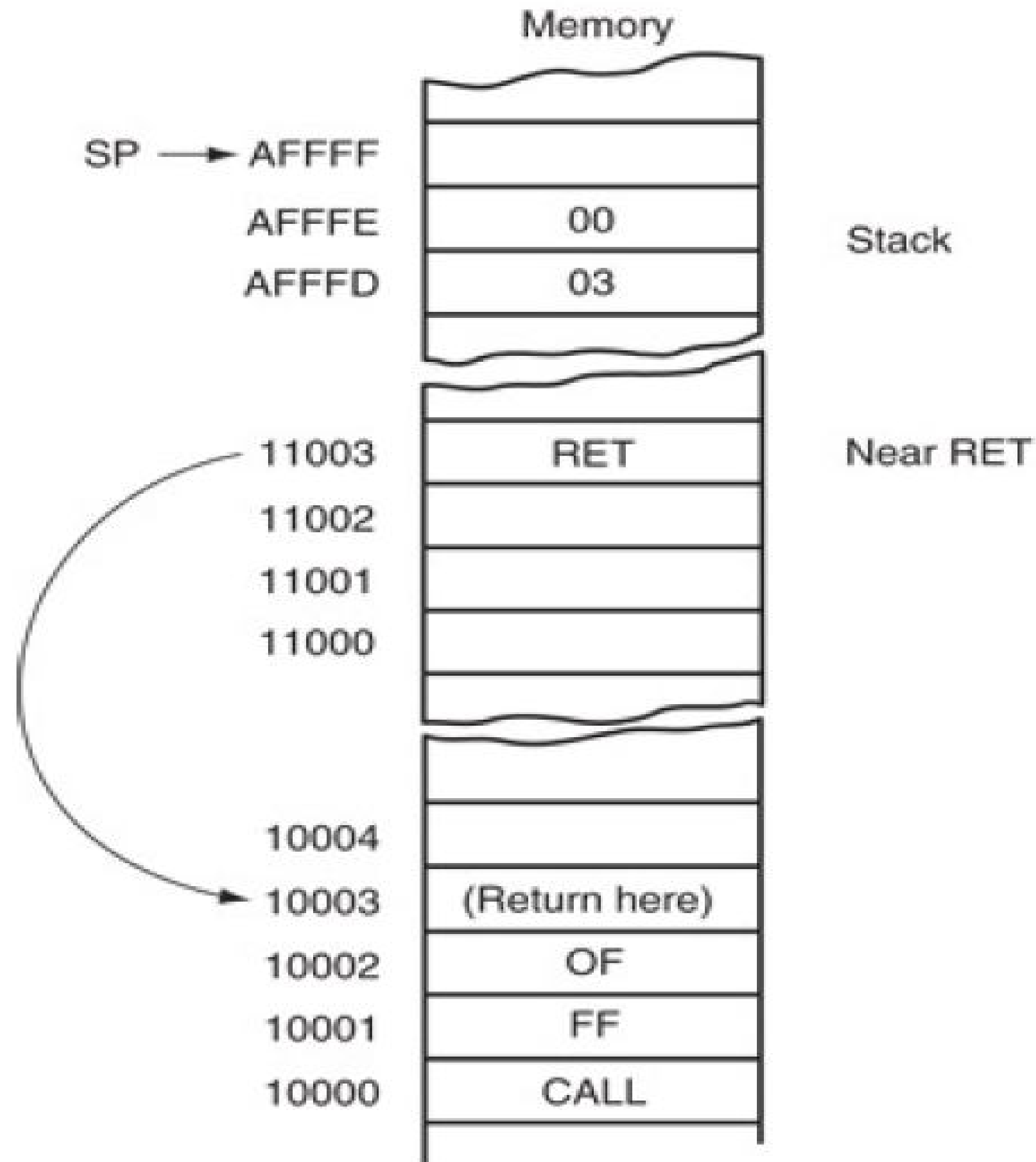
Copyright ©2009 by Pearson Education, Inc.
Upper Saddle River, New Jersey 07458 • All rights reserved.

RET

- Removes a 16-bit number or 32-bit number and places it in IP & CS.
- Figure 6–8 shows how the CALL instruction links to a procedure and how RET returns in the 8086–Core2 operating in the real mode.

Copyright ©2009 by Pearson Education, Inc.
Upper Saddle River, New Jersey 07458 • All rights reserved.

Figure 6–8 The effect of a return instruction on the stack and instruction pointer.



PEARSON
The Intel Microprocessors: 8086/8088, 80186/80188, 80286, 80386, 80486 Pentium, Pentium Pro Processor, Pentium II, Pentium, 4, and Core2 with 64-bit Extensions Architecture, Programming, and Interfacing, Eighth Edition
Barry B. Brey

Copyright ©2009 by Pearson Education, Inc.
Upper Saddle River, New Jersey 07458 • All rights reserved.