BITS Pilani

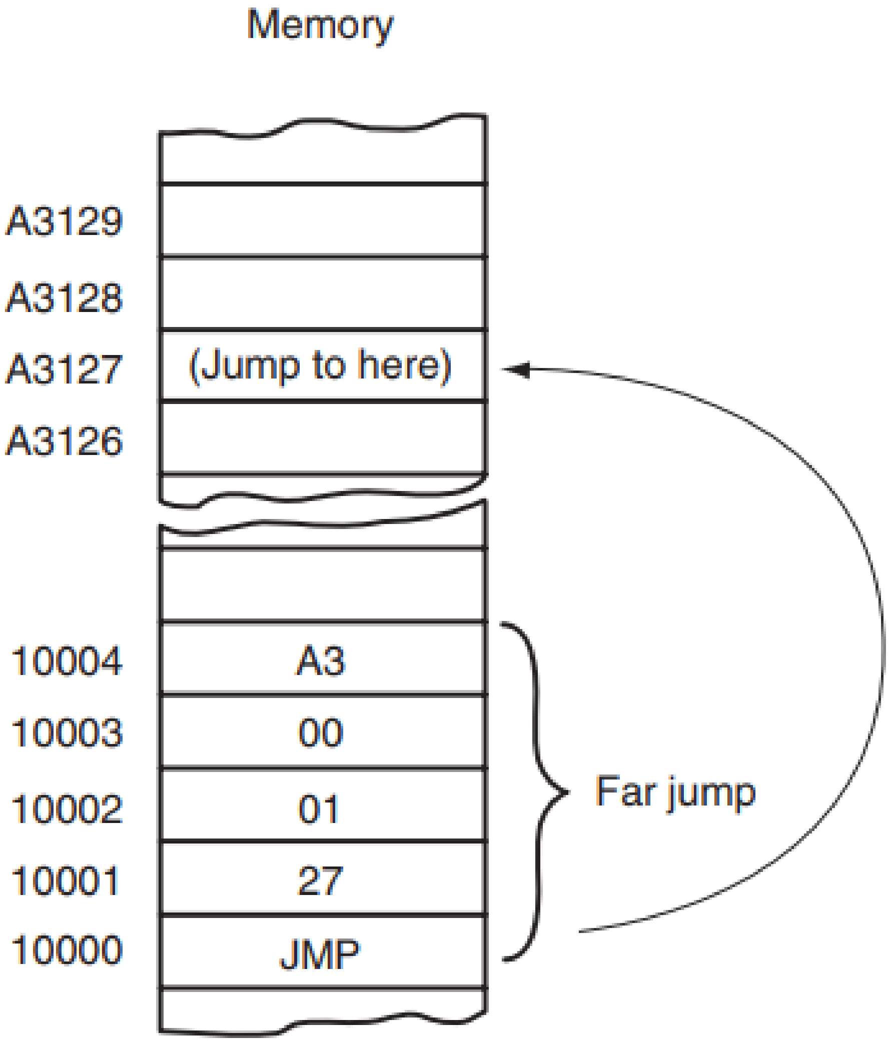
Microprocessors & Interfacing

INSTRUCTION SET

Dr. Gargi Prabhu
Department of CS & IS

Far Jump

```
                         EXTRN   UP:FAR

0000 33 DB                XOR BX,BX
0002 B8 0001      START: ADD AX,1
0005 E9 0200 R            JMP NEXT

                  ;<skipped memory locations>

0200 8B D8        NEXT:   MOV BX,AX
0202 EA 0002 —— R         JMP FAR PTR START

0207 EA 0000 —— R         JMP UP
```



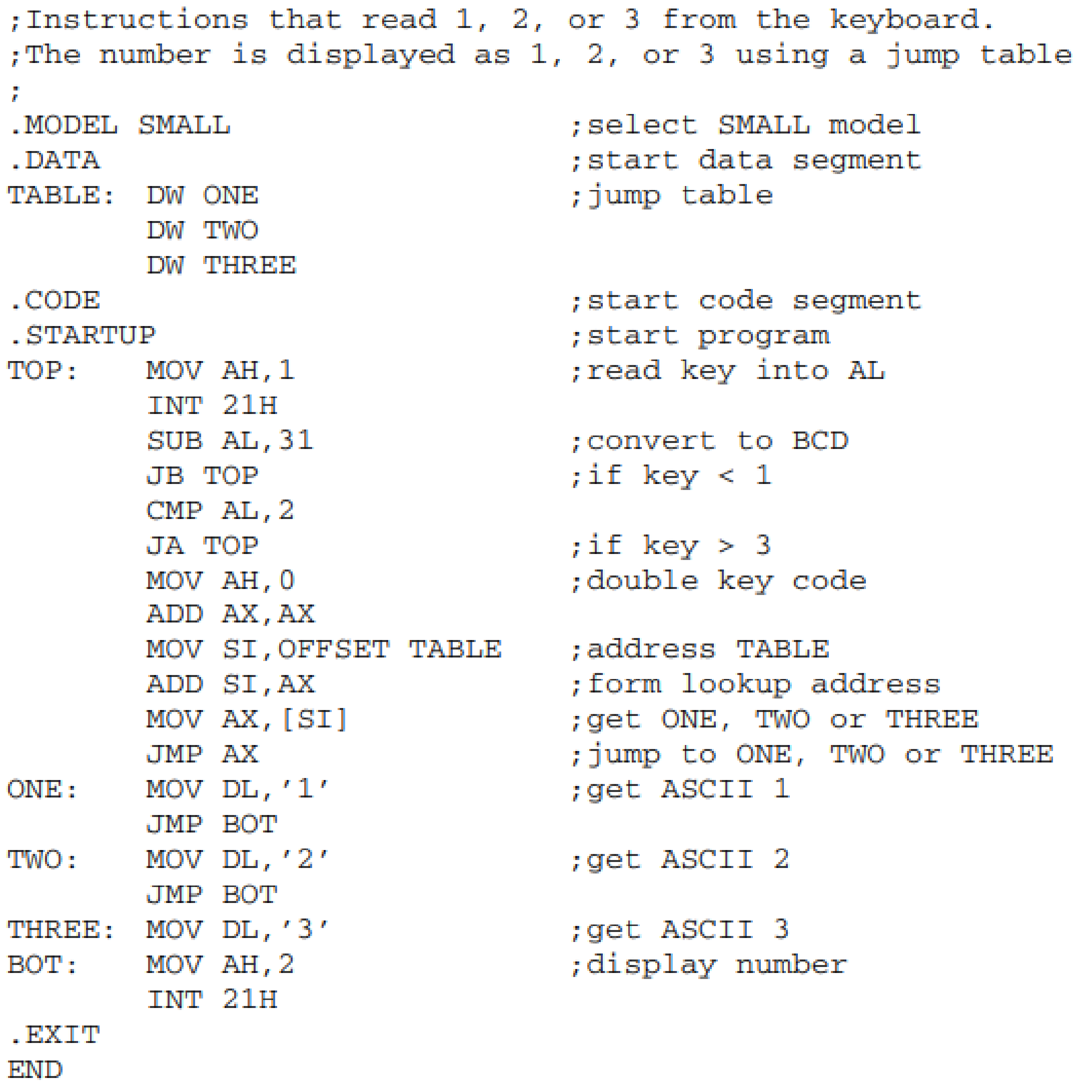
Jumps with Register Operands

- The jump instruction can also use a 16- or 32-bit register as an operand

E.g. JMP AX

Copies the contents of the AX register into the IP when the jump occurs.

Example



```
;Instructions that read 1, 2, or 3 from the keyboard.
;The number is displayed as 1, 2, or 3 using a jump table
;
.MODEL SMALL                  ;select SMALL model
.DATA                         ;start data segment
TABLE:  DW ONE                ;jump table
        DW TWO
        DW THREE
.CODE                         ;start code segment
.STARTUP                      ;start program
TOP:    MOV AH,1              ;read key into AL
        INT 21H
        SUB AL,31             ;convert to BCD
        JB TOP                ;if key < 1
        CMP AL,2
        JA TOP                ;if key > 3
        MOV AH,0              ;double key code
        ADD AX,AX
        MOV SI,OFFSET TABLE   ;address TABLE
        ADD SI,AX             ;form lookup address
        MOV AX,[SI]           ;get ONE, TWO or THREE
        JMP AX                ;jump to ONE, TWO or THREE
ONE:    MOV DL,’1’            ;get ASCII 1
        JMP BOT
TWO:    MOV DL,’2’            ;get ASCII 2
        JMP BOT
THREE:  MOV DL,’3’            ;get ASCII 3
BOT:    MOV AH,2              ;display number
        INT 21H
.EXIT
END
```

Indirect Jumps Using an Index

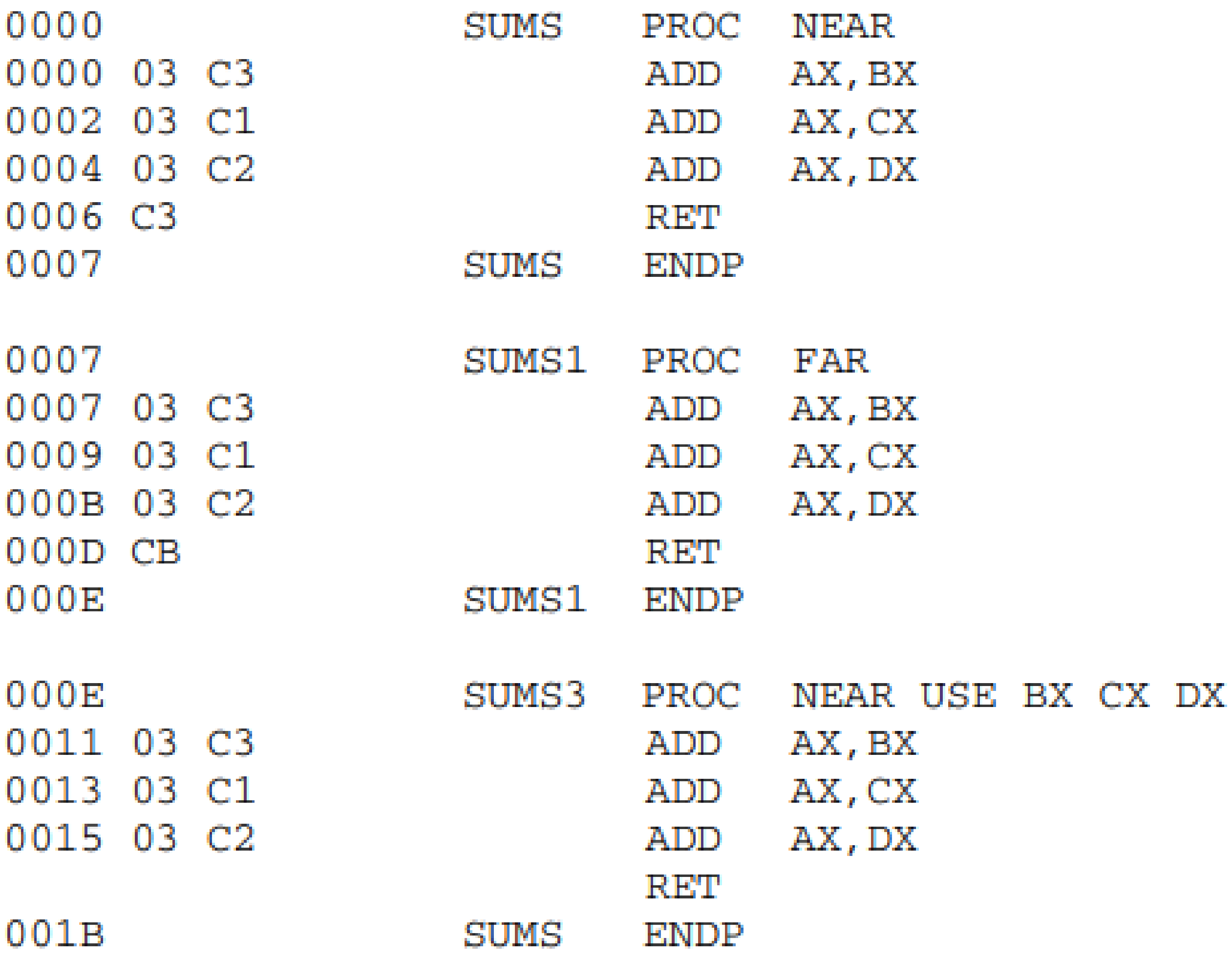
- The jump instruction may also use the [ ] form of addressing to directly access the jump table.

E.g. JMP TABLE [SI]

Procedures

- A procedure is a reusable section of the software that is stored in memory once, but used as often as necessary.
- The CALL instruction links to the procedure, and the RET (return) instruction returns from the procedure.
- The stack stores the return address whenever a procedure is called during the execution of a program.
- The CALL instruction pushes the address of the instruction following the CALL (return address) on the stack.
- The RET instruction removes an address from the stack so the program returns to the instruction following the CALL
- A procedure begins with the PROC directive and ends with the ENDP directive.

Example

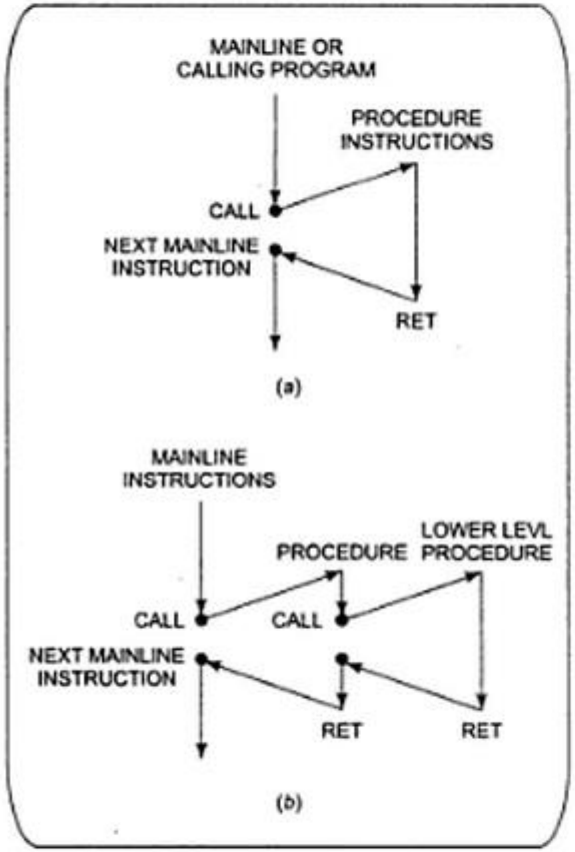


```
0000                  SUMS    PROC   NEAR
0000  03  C3                  ADD    AX,BX
0002  03  C1                  ADD    AX,CX
0004  03  C2                  ADD    AX,DX
0006  C3                      RET
0007                  SUMS    ENDP

0007                  SUMS1   PROC   FAR
0007  03  C3                  ADD    AX,BX
0009  03  C1                  ADD    AX,CX
000B  03  C2                  ADD    AX,DX
000D  CB                      RET
000E                  SUMS1   ENDP

000E                  SUMS3   PROC   NEAR USE BX CX DX
0011  03  C3                  ADD    AX,BX
0013  03  C1                  ADD    AX,CX
0015  03  C2                  ADD    AX,DX
                              RET
001B                  SUMS    ENDP
```

Call

- The 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.

Call & Return


MAINLINE OR CALLING PROGRAM
PROCEDURE INSTRUCTIONS
CALL
NEXT MAINLINE INSTRUCTION
RET
(a)
MAINLINE INSTRUCTIONS
PROCEDURE
LOWER LEVL PROCEDURE
CALL
CALL
NEXT MAINLINE INSTRUCTION
RET
RET
(b)

Types of CALL

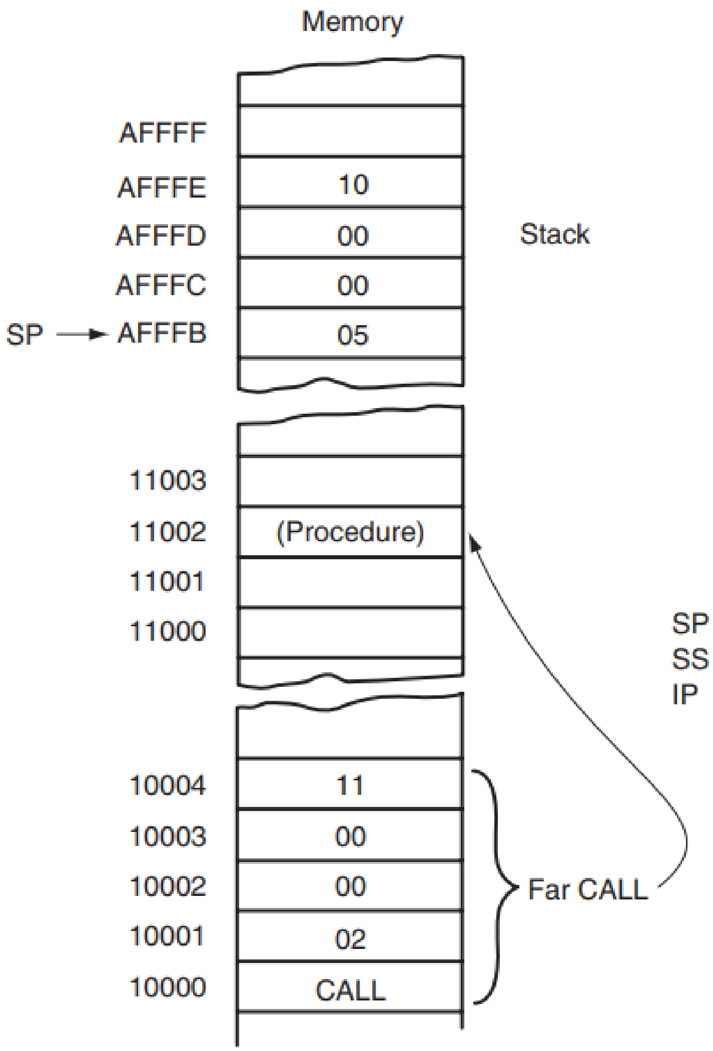
Near CALL

- Near CALL instruction is 3 bytes long; the first byte contains the opcode, and the second and third bytes contain the displacement, or distance of ±32K

Far CALL

- 5-byte instruction that contains an opcode followed by the next value for the IP and CS registers. Bytes 2 and 3 contain the new contents of the IP, and bytes 4 and 5 contain the new contents for CS.

Far CALL



SP before CALL = FFFF
SS before CALL = A000
IP before CALL = 0005

Types of Calls

CALLs with Register Operands

- contains a register operand.

E.g. CALL BX instruction, which pushes the contents of IP onto the stack. It then jumps to the offset address, located in register BX, in the current code segment

CALLs with Indirect Memory Addresses

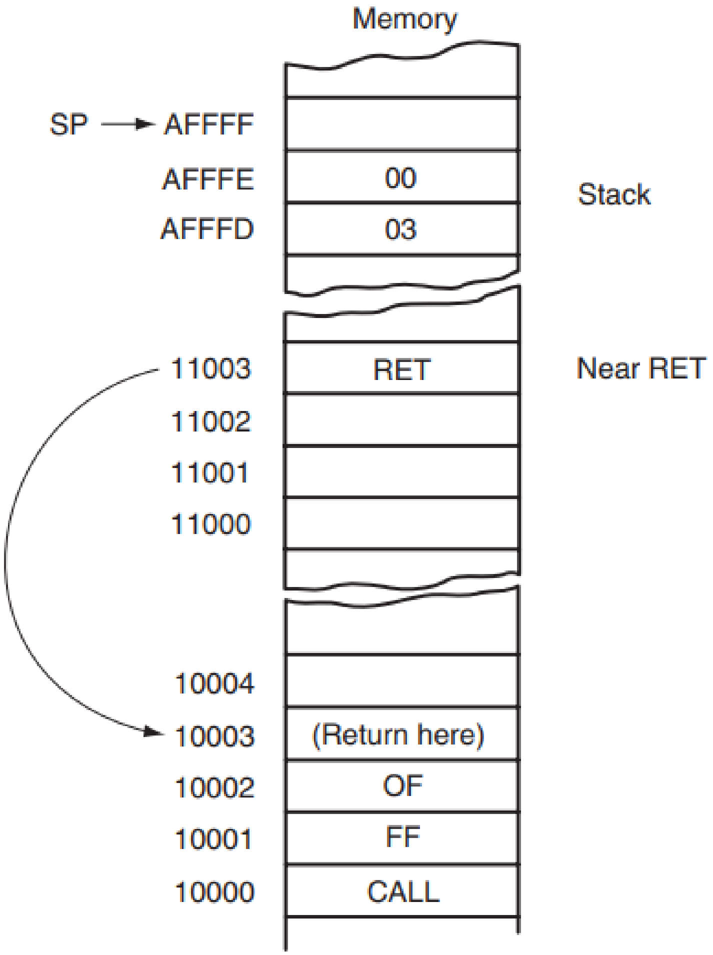
- Useful whenever different subroutines need to be chosen in a program.

E.g. CALL TABLE [4*EBX]

RET

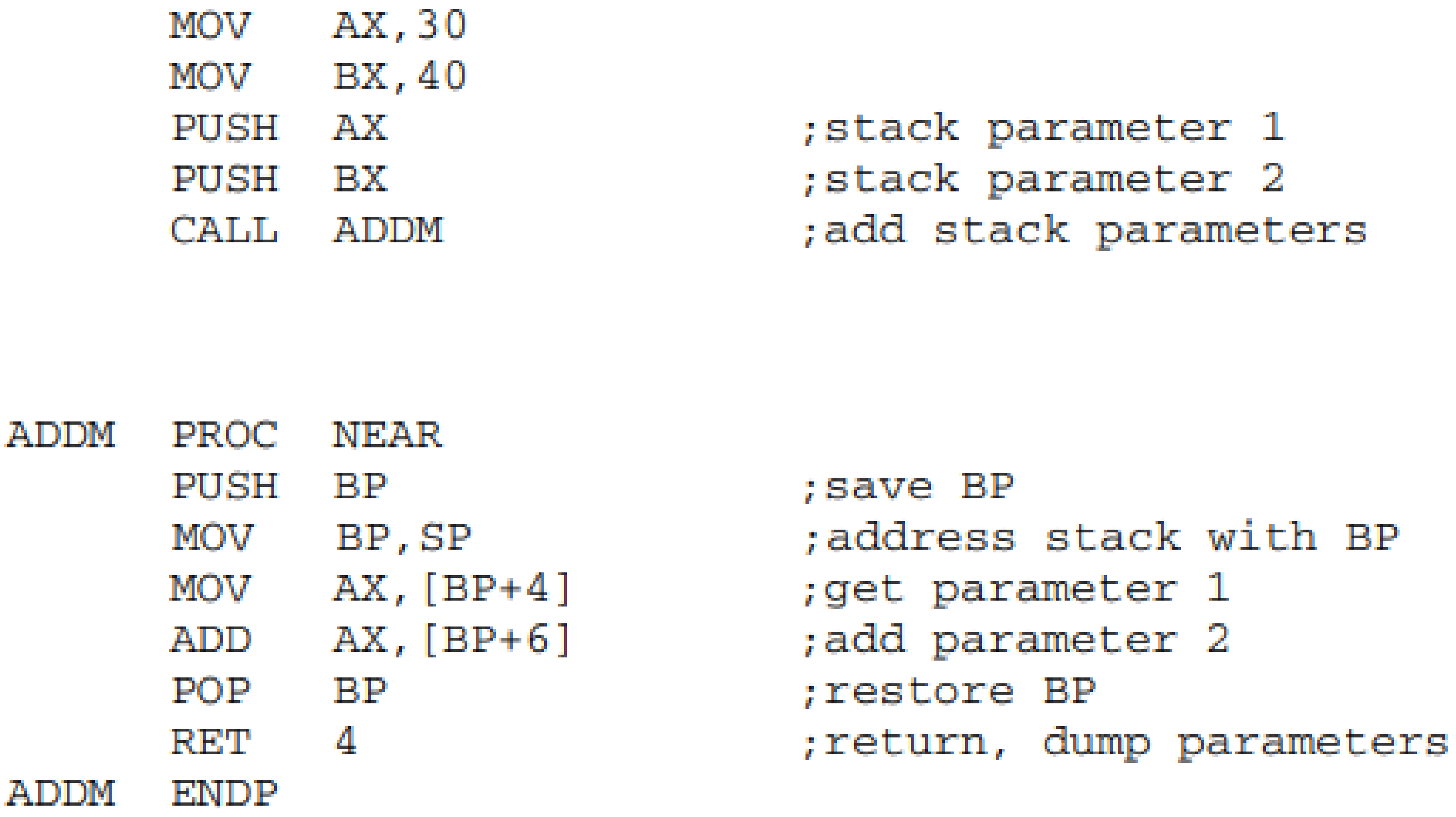
- The return instruction (RET) removes a 16-bit number (near return) from the stack and places it into IP, or removes a 32-bit number (far return) and places it into IP and CS.
- The near and far return instructions are both defined in the procedure’s PROC directive, which automatically selects the proper return instruction.

RET



SP before CALL = FFFD
SS before CALL = A000
IP before CALL = 1004

Example



```
      MOV   AX,30
      MOV   BX,40
      PUSH  AX               ;stack parameter 1
      PUSH  BX               ;stack parameter 2
      CALL  ADDM             ;add stack parameters


ADDM  PROC  NEAR
      PUSH  BP               ;save BP
      MOV   BP,SP            ;address stack with BP
      MOV   AX,[BP+4]        ;get parameter 1
      ADD   AX,[BP+6]        ;add parameter 2
      POP   BP               ;restore BP
      RET   4                ;return, dump parameters
ADDM  ENDP
```

Interrupts

- Hardware-generated CALL (externally derived from a hardware signal) OR
- Software-generated CALL (internally derived from the execution of an instruction or by some other internal event)
- At times, an internal interrupt is called an exception.

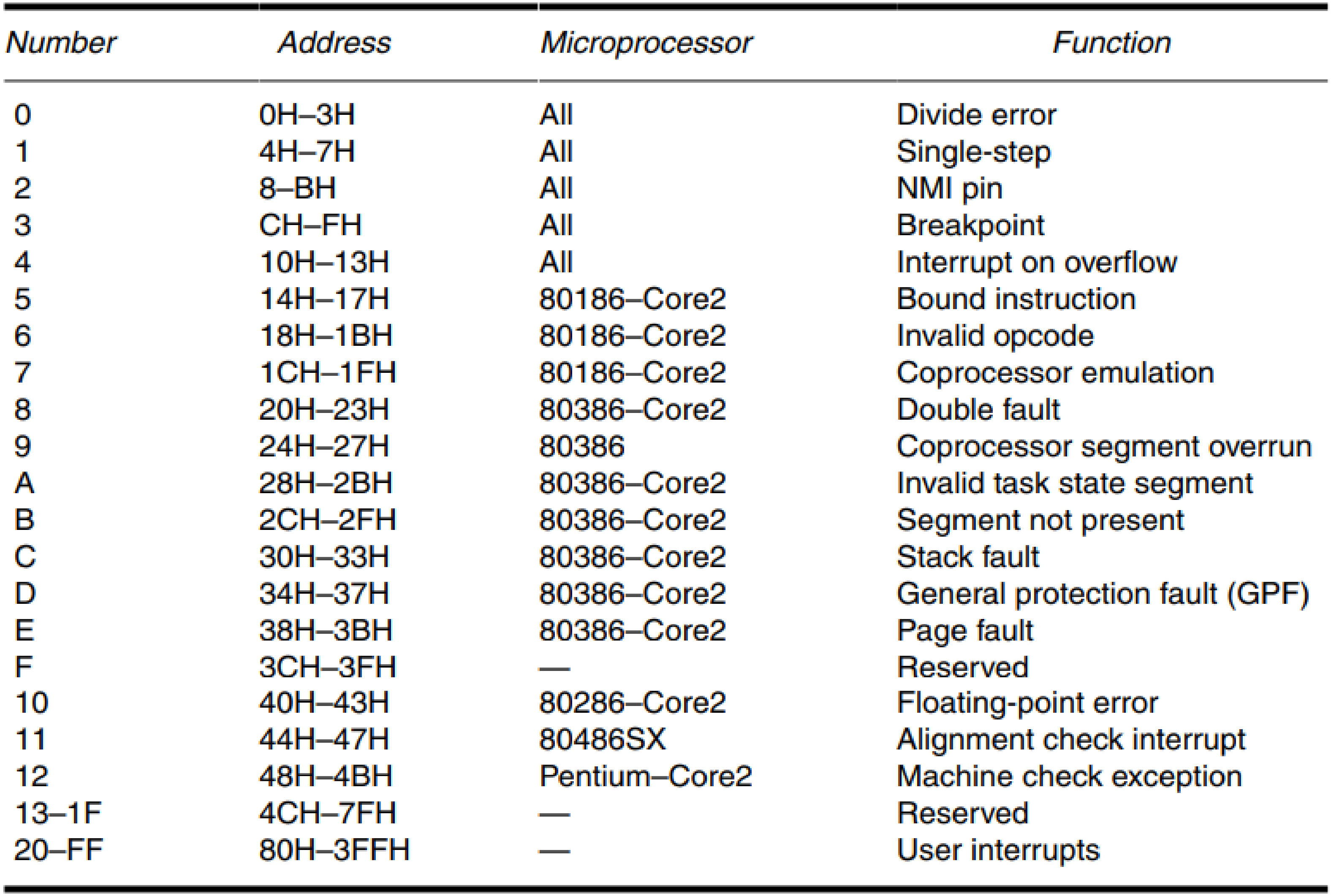
- Either type interrupts the program by calling an interrupt service procedure (ISP) or interrupt handler

Interrupt Vectors

- A 4-byte number stored in the first 1024 bytes of the memory (00000H–003FFH) when the microprocessor operates in the real mode.
- In the protected mode, the vector table is replaced by an interrupt descriptor table that uses 8-byte descriptors to describe each of the interrupts.
- There are 256 different interrupt vectors, and each vector contains the address of an interrupt service procedure.

- Each vector contains a value for IP and CS that forms the address of the interrupt service procedure. The first 2 bytes contain the IP, and the last 2 bytes contain the CS

Interrupt vectors defined by Intel



Number	Address	Microprocessor	Function
0	0H–3H	All	Divide error
1	4H–7H	All	Single-step
2	8–BH	All	NMI pin
3	CH–FH	All	Breakpoint
4	10H–13H	All	Interrupt on overflow
5	14H–17H	80186–Core2	Bound instruction
6	18H–1BH	80186–Core2	Invalid opcode
7	1CH–1FH	80186–Core2	Coprocessor emulation
8	20H–23H	80386–Core2	Double fault
9	24H–27H	80386	Coprocessor segment overrun
A	28H–2BH	80386–Core2	Invalid task state segment
B	2CH–2FH	80386–Core2	Segment not present
C	30H–33H	80386–Core2	Stack fault
D	34H–37H	80386–Core2	General protection fault (GPF)
E	38H–3BH	80386–Core2	Page fault
F	3CH–3FH	—	Reserved
10	40H–43H	80286–Core2	Floating-point error
11	44H–47H	80486SX	Alignment check interrupt
12	48H–4BH	Pentium–Core2	Machine check exception
13–1F	4CH–7FH	—	Reserved
20–FF	80H–3FFH	—	User interrupts

innovate achieve lead

BITS Pilani

Pilani Campus

Thank You