① MOV - MOV Destination, Source :-

* MOV CX, 037 H ; Put 037H to CX
* MOV CX, BX ; Copy content of BX to CX
* MOV BL, [437H] ; Copy byte in DS at offset 437H to BL

② LEA - LEA Register, Source :-

* LEA BX, PRICE ; Load BX with offset of PRICE in DS
* LEA BP, SS : STACK_TOP ; Load BP with offset of STACK_TOP in SS
* LEA CX, [BX][DI] ; Load CX with EA = [BX]+[DI]

③ ADD - ADD Destination, Source
ADC - ADC Destination, Source

* ADD AL, 01H ; Add immediate number 01H to content of AL

*ADD DX, BX ; Add content of BX to content of CX

*ADD DX, [SI] ; Add word from memory at offset [SI] in DS to content of DX

*ADC AL, BL ; Add content of BL plus carry status to content of CL

*ADC AL, PRICES [BX] ; Add byte from effective addre PRICE [BX] plus carry statu to content of AL

④ SUB - SUB

SBB - SBB

*SUB CX, BX ; Sub content of BX from CX.

*SUB AX, 01H ; Sub immediate number 01H from content of AX

*SUB PRICES [BX], 04H ; Sub 04 from effective address PRICES [BX]

*SBB CH, AL ; Sub content of AL and CF from content of CH.

*SBB BX, 01H ; Sub immediate number 01H from content of AX

*SBB CX, TABLE [BX]; Sub word from effective address from TABLE[BX] and status of CF from CX.

⑤ MUL - MUL Source :-

*MOV AL, MCAND_8

MUL BH ; MUL AL with BH, result in AX

*MOV AX, MCAND_16

MOV CL, MPLIER_8

MOV CH, 00H

MUL CX ; MUL AX with CX, 32 bit result store in DX, AX

⑥ DIV - DIV Source :-

*MOV AX, DDEND_16

DIV BL ; Div word in AX by byte in BL, Quotient in AL, remainder in AH

*MOV DX, DDEND_16

MOV AX, DDEND_16

DIV CX ; div word in DX and AX by word in CX, Quotient in AX, Remainder in DX

⑦ INC - INC Destination :-

*INC BL ; Add 1 to content of BL register

*INC BYTE PTR [BX] ; Add 1 to byte in data segment at offset contained in BX

⑧ DEC - DEC destination :-

*DEC CL ; Subtract 1 from content of CL reg

*DEC BYTE PTR [BX] ; Subtract 1 from byte at offset [BX] in DS.

*DEC COUNT ; Subtract 1 for byte or word named COUNT in DS. Decrement a byte if COUNT is declared with a DB. Decrement a word if COUNT is declared with a DW.

⑨ DAA (Decimal Adjust After BCD Addition):-

*Let AL = 59 BCD, and BL = 35 BCD

ADD AL, BL ; AL = 8EH, lower nibble > 9, add
DAA ; 06H to AL
AL = 94 BCD, CF = 0

* Let AL = 88 BCD, and BL = 49 BCD

ADD AL, BL ; AL = D1H, AF = 1, add 06H to AL
DAA AL = D7H, upper nibble > 9

add 60H to AL

AL = 37 BCD, CF = 1

⑩ AAA (ASCH Adjust for Addition) :-

* Let AL = 00110101 (ASCII 5), BL = 00111001 (ASCII 9).

ADD AL, BL ; AL = 01101110 (6EH, which is incorrect BCD)

AAA

AL = 00000100 (unpacked BCD 4

CF = 1 indicates answer is 14 decimal.

⑪ AND - AND Destination, Source :-

* AND CX, [SI] ; AND word in DS at offset with word in CX register

* AND BH, CL ; AND byte in CL with byte in BH.

* AND BX, 00FFH ; 00FFH masks upper bytes, leaves lower bytes unchanged

⑫ OR-OR Destination-Source :-

* OR AH, CL ; CL ORed with AH, result in AH, CL not changed
* OR BL, 80H ; BL ORed with immediate number 80H, sets MSB of BL to 1
* OR CX, TABLE [SI] ; CX ORed with word from effective address TABLE[SI], content of memory is not change

⑬ XOR-XOR Destination, Source :-

* XOR CL, BH ; Byte in BH exclusive-ORed with byte in CL.
* XOR WORD PTR [BX], 00FFH ; Exclusive-OR immediate number 00FFH with word at offset [BX] in the data segment, result in memory location [BX]

(14) CMP - CMP Destination, Source :-

CX = BX ; Result of subtraction is 0

CX > BX ; No borrow required, so CF = 0

CX < BX ; Subtraction requires borrow, so CF = 1

* CMP AL, 01H ; compare immediate number 01H with byte in AL.

* CMP BH, CL ; compare byte in CL with byte in BH

* CMP CX, TEMP ; compare word in DS at displacement TEMP with word at CX

* CMP PRICES[BX], 49H ; compare immediate number 49H, with byte at offset [BX]

(15) TEST - TEST Destination, Source :-

* TEST AL, BH ; AND BH with AL, no result store

* TEST CX, 0001H ; AND CX with immediate number 0001H

* TEST BP, [BX], [DI] ; AND word are offset [BX][DI] in DS with word in BP, no result stored.

⑥ RCL - RCL Destination, Count :-

* RCL DX, 1 ; word in DX 1 bit left, MSB to CF, CF to LSB.

* MOV CL; 4 ; Load the number of bit position
RCL SUM [BX], CL ; Rotate byte or word at effective address SUM[BX] 4 bits left. Original bit 4 now in CF

⑰ RCR - RCR Destination, Count :-

* RCR BX, 1 ; word in BX right 1 bit, CF to MSB, LSB to CF

* MOV CL, 4 ; Load CL of bit position
RCR BYTE PTR [BX], 4 ; Rotate the byte at offset [BX] in DS 4bit positions right
CF - original bit 3

⑱ SAL - SAL Destination, Count

SHL - SHL Destination, Count

* SAL BX, 1 ; shift word in BX 1 bit positions left 0 in LSB

* MOV CL, 02H ; Load desired number

SAL BP, CL ; shift word in BP Left CL bit positions, 0 in LSBs

⑲ SAR - SAR Destination, Count

* SAR DX, 1 ;; shift word in DI one bit positi- on right, new MSB = OLD MSB

* MOV CL, 02H ; Load desired number

SAR WORD PTR [BP], CL ; shift word at offset [BP] in stack segment right by two bit positions.

⑨ SHR - SHR Destination, Count :-

```
SHR  BP, 1 ; shift word in BP one bit position
             right, 0 in MSB

MOV  CL, 03H ; Load desired number
SHR  BYTE PTR [BX]; shift byte in DS at offset[BX]
                    3 bits right, 0's in 3 Mbps
```

㉑ JBE / JNA (Jump if below or equal / Jump if not above) :-

```
* CMP  AX, 4371H ; compare (AX - 4371H)
  JBE  NEXT      ; Jump to label NEXT if AX is
                   below or equal to 4371H.

* CMP  AX, 4371H ; Compare (AX - 4371H)
  JNA  NEXT      ; Jump to label NEXT if AX not
                   above 4371H
```

(22) JG/ JNLE (Jump if greater / Jump if not less than or equal) :-

* CMP BL, 39H ; compare by subtract 39H from BL
  JG NEXT ; jump to label NEXT

* CMP BL, 39H ; compare by subtract 39H from BL
  JNLE NEXT ; Jump to label NEXT if BL is not less than or equal to 39H

(23) JL / JNGE (Jump if less than / Jump if not greater than or equal) :-

* CMP BL, 39H ; compare by subtract 39H from BL
  JL AGAIN ; Jump to label AGAIN if BL more negative than 39H

* CMP BL, 39H ;
  JNGE AGAIN ; Jump to label AGAIN if BL not more positive than or equal to 39H

(24) JLE / JNG (Jump if less than or equal / Jump if not greater) :-

* CMP BL, 39H ; compare by subtracting 39H from BL
  JLE NEXT ; Jump to label NEXT if BL mor negative than or equal to 39H

* CMP BL, 39H ; compare
  JNG NEXT ; Jump to label NEXT if BL not more positive than 39H

(25) JE / JZ (Jump if equal / Jump if zero) :-

* CMP BX, DX ; compare (BX - DX)
  JE DONE ; Jump to DONE if (BX-DX)

* IN AL, 30H ; read data from port 8FH
  SUB AL, 30H ; subtract the minimum value.
  JZ START ; Jump to label START if the result of subtraction is 0.

(26) JNE / JNZ (Jump not equal / Jump if not zero):

* IN AL, 0F8H ; read data
  CMP AL, 72 ; compare (AL - 72)
  JNE NEXT ; Jump to label NEXT if (AL ≠ 72)

* ADD AX, 0002H ; add count factor 0002H to AX
  DEC BX ; decrement BX
  JNZ NEXT ; Jump to label NEXT if BX ≠ 0

(27) PUSH - PUSH Source :-

* PUSH BX ; decrement SP by 2, copy BX to stack

* PUSH TABLE[BX] ; decrement SP by 2, copy word from memory in DS at EA = TABLE + [BX] to stack

(28) POP - POP Destination :-

* POP DX ; copy a word from top of stack to DX , increment SP by 2
* POP TABLE [DX] ; copy a word from top of stack to memory in DS with EA = TABLE + [BX] , increment SP by 2

(29) IN - IN Accumulator, Port :-

* IN AL , 0C8H ; input a byte from port 0C8H to AL
* MOV DX , 0FF78H ; initialize DX to point to port
  IN AL , DX ; input a byte from 8bit port 0FF78H to AL

(30) OUT - OUT Port, Accumulator :-

* OUT 3BH , AL ; copy the content of AL to port
* MOV DX , 0FF78H ; Load desired port address
  OUT DX , AL ; Copy content of AL to port 0FF78H