8086 ASSEMBLY LANGUAGE PROGRAMS(FOR THEORY ONLY)

In Assembly Language Program(ALP) , we use three accumulators, one is AL for 8-bit operation, AX for 16-bit operation. DX is for 32 bit operations if the result or output exceed 16-bits. It is used in multiplication and division.

In ALP we use two pointers, one is SI and another is DI. The SI stands for source index and DI for destination index. The SI and DI are used in comparisons , sorting, biggest and smallest, block transfer, multibyte etc… But in most programs either SI or DI is used.

Write an Assembly Language Program(ALP) to find the smallest of n numbers

LABEL	MNEMONICS

	MOV CL,04
	MOV SI,1000
	MOV AL,[SI]

GO :	INC SI
	CMP AL,[SI]
	JC NEXT
	MOV AL,[SI]

NEXT :	DEC CL JNL GO INC SI MOV [SI] , AL HLT

\

Write an ALP to find the biggest of n numbers

LABEL	MNEMONICS

	MOV CL,04
	MOV SI,1000
	MOV AL,[SI]

GO :	INC SI
	CMP AL,[SI]
	JNC NEXT
	MOV AL,[SI]

NEXT :	DEC CL JNL GO INC SI MOV [SI] , AL HLT

Write an ALP to sort the numbers in Ascending Order

LABEL	MNEMONICS
	MOV DL,04
AGAIN :	MOV SI,1000 MOV CL,04

GO:	MOV AL,[SI]
	INC SI
	MOV BL,[SI]
	CMP AL, BL
	JC NEXT
	MOV [SI],AL
	DEC SI
	MOV [SI],BL
	INC SI

NEXT :	DEC CL JNZ GO DEC DL JNL AGAIN HLT

Write an ALP to arrange the numbers in Descending order

LABEL	MNEMONICS
	MOV DL,04
AGAIN :	MOV SI,1000
	MOV CL,04
GO:	MOV AL,[SI]
	INC SI
	MOV BL,[SI]
	CMP AL, BL
	JNC NEXT
	MOV [SI],AL
	DEC SI
	MOV [SI],BL
	INC SI
NEXT :	DEC CL
	JNZ GO
	DEC DL
	JNL AGAIN
	HLT

Write an ALP to compare two strings

LABEL	MNEMONICS
	MOV CL,05
	MOV SI,1000
	MOV DI,2000
	MOV BX,3000
NEXT :	CMPS B
	JNZ NEQUAL
	DEC CL
	JNZ NEXT
	MOV [BX],01
	JMP LAST
NEQUAL :	MOV [BX],00
LAST :	HLT

01 in location 3000 indicates equal
00 in location 3000 indicates not equal

Write an ALP to perform multibyte addition

LABEL	MNEMONICS

	CLC
	MOV SI,1000
	MOV DI,2000
	MOV CL,05

NEXT :	MOV AL,[SI]
	ADD [DI] , AL
	INC SI
	INC DI
	DEC CL
	JNZ NEXT
	JNC LAST
	MOV [DI],01

LAST :	HLT

Write and ALP to perform multibyte subtraction

LABEL	MNEMONICS

	CLC
	MOV SI,1000
	MOV DI,2000
	MOV CL,05

NEXT :	MOV AL,[SI] SUB [DI] , AL INC SI INC DI DEC CL JNZ NEXT
LAST :	HLT

Write an ALP to perform 8bit division

LABEL	MNEMONICS

	MOV SI,1000
	MOV AL, [SI]
	INC SI
	MOV BL,[SI]
	DIV BL
	INC SI
	MOV [SI], AX
	HLT

Write an ALP to perform 16-bit division

LABEL	MNEMONICS

	MOV SI,1000
	MOV AX, [SI]
	INC SI
	INC SI
	MOV BX,[SI]
	DIV BX
	INC SI
	INC SI
	MOV [SI],AX
	INC SI
	INC SI
	MOV [SI],DX
	HLT

Write an ALP to perform 16-bit addition

LABEL	MNEMONICS

	MOV SI,1000
	MOV AX, [SI]
	INC SI
	INC SI
	MOV BX,[SI]
	ADD AX,BX
	INC SI
	INC SI
	MOV [SI],AX
	HLT

Write an ALP to perform 16-bit subtraction

LABEL	MNEMONICS

	MOV SI,1000
	MOV AX, [SI]
	INC SI
	INC SI
	MOV BX,[SI]
	SUB AX,BX
	INC SI
	INC SI
	MOV [SI],AX
	HLT

Write an ALP to perform 8-bit multiplication

LABEL	MNEMONICS

	MOV SI,1000
	MOV AL,[SI]
	INC SI
	MOV BL,[SI]
	MUL BL
	INC SI
	MOV [SI],AX
	HLT

Write an ALP to perform 16-bit multiplication

LABEL	MNEMONICS

	MOV SI,1000
	MOV AX,[SI]
	INC SI
	INC SI
	MOV BX,[SI]
	MUL BX
	INC SI
	INC SI
	MOV [SI],AX
	INC SI
	INC SI
	MOV [SI],DX
	HLT

SUM OF SERIES (8-bit)

LABEL	MNEMONICS

NEXT :	MOV CL,05
	MOV SI,1000
	MOV AL,00
	ADD AL, [SI]
	INC SI
	DEC CL
	JNZ NEXT
	MOV [SI],AL
	HLT

Write an ALP to perform block data transfer

LABEL	MNEMONICS

NEXT :	MOV CL,05
	MOV SI,1000
	MOV DI,2000
	MOV AL,[SI]
	MOV [DI],AL
	INC SI
	INC DI
	DEC CL
	JNZ NEXT
	HLT

Write an ALP to perform sum of series of 16-bit number

LABEL	MNEMONICS

NEXT :	MOV CL,05
	MOV SI,1000
	MOV AX,0000
	ADD AX,[SI]
	INC SI
	INC SI
	DEC CL
	JNZ NEXT
	MOV [SI],AX
	HLT

Write an ALP to perform 16-bit block data transfer

LABEL	MNEMONICS

NEXT :	MOV CL,05
	MOV SI,1000
	MOV DI,2000
	MOV AX,[SI]
	MOV [DI],AX
	INC SI
	INC SI
	INC DI
	INC DI
	DEC CL
	JNZ NEXT
	HLT

Write an ALP to find the length of the string

LABEL	MNEMONICS

	MOV SI,1000
	MOV CL,00
	MOV AL, '*'

NEXT :	CMP AL,[SI] JZ LAST INC CL INC SI JMP NEXT
LAST :	INC SI MOV [SI],CL HLT

Write an ALP to find the number of occurrences of a string

LABEL	MNEMONICS

	MOV CL,05
	MOV DL,00
	MOV SI,1000
	MOV AL, 'A'

GO :	CMP AL, [SI] JNZ NEXT INC CL
NEXT :	INC SI DEC CLS JNZ GO MOV [SI],DL HLT

Write an ALP to perform 8-bit addition

LABEL	MNEMONICS

	MOV DI,1000H
	MOV AL,[DI]
	INC DI
	MOV BL,[DI]
	ADD AL,BL
	INC DI
	MOV [DI],AL
	HLT

Write an ALP to perform 8-bit subtraction

LABEL	MNEMONICS

	MOV DI,1000H
	MOV AL,[DI]
	INC DI
	MOV BL,[DI]
	SUB AL,BL
	INC DI
	MOV [DI],AL
	HLT

Write an ALP to find ONE'S complement of 8 –bit number

LABEL	MNEMONICS

	MOV DI,1000H
	MOV AL,[DI]
	NOT AL
	INC DI
	MOV [DI],AL
	HLT

Write an ALP to find TWO'S complement of 8 –bit number

LABEL	MNEMONICS

	MOV DI,1000H
	MOV AL,[DI]
	NEG AL(OPERAND IS OPTIONAL)
	INC DI
	MOV [DI], AL
	HLT

NOTE

'NOT' - PERFORMS ONE'S COMPLEMENT

'NEG' - STANDS FOR NEGATE OPERATION WHICH PERFORMS TWO'S COMPLEMENT

IN GENERAL NEG HAS NO OPERAND , BUT FOR PRACTICAL PURPOSE IT REQUIRES THE OPERAND

NEG HAS OPERAND IN SOME CASES, THE OPERAND IS OPTIONAL FOR NEG

Write an ALP to find ONE'S complement of 16 –bit number

LABEL	MNEMONICS

	MOV DI,1000H
	MOV AX,[DI]
	NOT AX
	INC DI
	INC DI
	MOV [DI],AX
	HLT

Write an ALP to find TWO'S complement of 16 –bit number

LABEL	MNEMONICS

	MOV DI,1000H
	MOV AX,[DI]
	NEG AX(OPERAND IS OPTIONAL)
	INC DI
	INC DI
	MOV [DI] , AX
	HLT

MASKING LSB 4-BITS OF 8-BIT NUMBER

(Masking is a process that converts the group of bits or portion of bits of a number into zero. Masking is known as clear operation).

LABEL	MNEMONICS

	MOV DI,1000H
	MOV AL,[DI]
	AND AL,F0H
	INC DI
	MOV [DI], AL
	HLT

Example : Operand in AL is 34H , to mask the LSB of 34H to zero, we perform the AND operation with 34H by a value F0H to produce the output as 30H. Here in our example 34H is a 8bit number, the LSB of a number here is 4 and we change 4 to zero as follows

```
34H   =     0011 0100
F0H   =     1111 0000
          ------------------
30H         0011 0000 (OUTPUT)
          ------------------
```

MASKING MSB 4-BITS OF 8-BIT NUMBER

LABEL	MNEMONICS

	MOV DI,1000H
	MOV AL,[DI]
	AND AL,0FH
	INC DI
	MOV [DI], AL
	HLT

Example : Example : Operand in AL is 34H , to mask the MSB of 34H to zero, we perform the AND operation with 34H by a value 0FH to produce the output as 04H. Here in our example 34H is a 8bit number, the MSB of a number here is 3 and we change value 3 to zero as follows

```
34H   =     0011 0100
0FH   =     0000 1111 (AND OPERATION)
          ------------------
04H         0000 0100 (OUTPUT)
          ------------------
```

MASKING LSB 8-BITS OF A 16-BIT NUMBER

LABEL	MNEMONICS

	MOV DI,1000H
	MOV AX,[DI]
	AND AX,FF00H
	INC DI
	INC DI
	MOV [DI],AX
	HLT

Example : Mask the LSB of the number 1234H

Method :

```
1234H =  0001 0010  0011  0100 (Input Value)
FF00H = 1111 1111  0000  0000  (Operator)
          ---------------------------------
1200H     0001 0010 0000  0000   (output)
          ---------------------------------
```

MASKING MSB 8-BITS OF A 16-BIT NUMBER

LABEL	MNEMONICS

	MOV DI,1000H
	MOV AX,[DI]
	AND AX,00FFH
	INC DI
	INC DI
	MOV [DI],AX
	HLT

Example : Mask the LSB of the number 1234H

Method : 1234H = 0001 0010 0011 0100 (Input Value)
00FFH = 0000 0000 1111 1111 (Operator)

0034H 0000 0000 0011 0100 (output)

SHIFT LEFT OF 8-BIT NUMBER

LABEL	MNEMONICS

	MOV DI,1000H
	MOV AL,[DI]
	SHL (or) SHL AL (or) SHL AL,01
	INC DI
	MOV [DI],AL
	HLT

In general SHL does not require an operand, it shifts the Accumulator value by left or right by one bit . But in performing the program using the assembler it requires the operand like accumulator with the shift value, for example SHL AL,01,which means shifting the accumulator value to left by one bit.

Example : If we want to shift left the value 12H by one bit left produces the different output.

Input : 12H = 0001 0010

By shifting the above binary value to left by one bit the value becomes

0010 0100 = 24H

SHIFT RIGHT OF 8-BIT NUMBER

LABEL	MNEMONICS

	MOV DI,1000H
	MOV AL,[DI]
	SHR (or) SHR AR (or) SHL AR, 01
	INC DI
	MOV [DI],AL
	HLT

Example : If we want to shift the value 12H by one bit right produces the different output.

Input : 12H = 0001 0010

By shifting the above binary value to right by one bit the value becomes

0000 1001 = 09H

SHIFT LEFT OF 16-BIT NUMBER

LABEL	MNEMONICS

	MOV DI,1000H
	MOV AX, [DI]
	SHL (or) SHL AX (or) SHL AX,01
	INC DI
	INC DI
	MOV [DI],AX
	HLT

Example:

Input value: 1234 H

By shifting the value 1234H to shift left changes the value

Input: 1234H = 0001 0010 0011 0100

By shifting the value by one bit left the value becomes

= 0010 0100 0110 1000(2468)

Hence the output after shift left of the value of 1234H becomes 2468H

SHIFT RIGHT OF A 16-BIT NUMBER

LABEL	MNEMONICS

	MOV DI,1000H
	MOV AX, [DI]
	SHR (or) SHR AX (OR) SHR AX, 01
	INC DI
	INC DI
	NOV [DI],AX
	HLT

Example:

Input value: 1234 H

By shifting the value 1234H to shift right changes the value

Input: 1234H = 0001 0010 0011 0100

By shifting the value by one bit right the value becomes

= 0000 1001 0001 1010(091A)

Hence the output after shift left of the value of 1234H becomes 091AH