12 / 12 / 2022

Experiment No: 8

SORT N-BIT NUMBERS USING 8086 TRAINER KIT

Aim

Write a program sort n bit numbers using trainer kit.

Program

Address	Mnemonics	Comment
2000	MOV SI, 1500	SI <- 1500
2004	MOV CL, [SI]	CL <- [SI]
2006	DEC CL	CL <- CL-1
2008	MOV SI, 1500	SI <- 1500
200C	MOV CH, [SI]	CH <- [SI]
200E	DEC CH	CH <- CH-1
2010	INC SI	SI <- SI +1
2011	MOV AL, [SI]	AL <- [SI]
2013	INC SI	SI <- SI +1
2014	CMP AL, [SI]	AL <- [SI]
2016	JC 201E	Jump to 201E if CY=1
2018	XCHG AL, [SI]	Swap AL and [SI]
201A	DEC SI	SI <- SI -1
201B	XCHG AL, [SI]	Swap AL and [SI]
201D	INC SI	SI <- SI +1

Address	Mnemonics	Comments
201E	DEC CH	CH <- CH-1
2020	JNZ 2011	Jump to 2011 ; if ZF = 0
2022	DEC CL	CL <- CL-1
2024	JNZ 2008	Jump to 2008 ; if ZF = 0
2026	HLT	END.

Result

The program to sort n bit numbers using 8086 trainer kit has been exicuted and the output is verified. It satisfies CO1.

INPUT

[1500] <- 04

[1501] <- 03

[1502] <- 06

[1503] <- 02

[1504] <- 05

Output

[1500] <- 04

[1500] <- 02

[1500] <- 03

[1500] <- 05

[1500] <- 06

19 / 12 / 2022

Experiment No: 9

STUDY OF ASSEMBLER AND DEBUGGING COMMANDS

Aim

Study of assembler and debugging commands.

Theory.

Debugging Commands

- It is a program to write, execute and debug assembly language programs, and to examine the content of registers and memory.
- To start DEBUG its name is typed after DOS prompt.
  ie C:\>DEBUG.
- A DEBUG command is represented by a one-letter symbol and are not case-sensitive.
- A - Assemble Command: It allows user to enter assembly language program and convert it into machine codes.
  Syntax : A [Offset-address]
- U - Unassemble Command: It disassembles machine codes of specified memory addresses and gives corresponding mnemonics.
  Syntax: U [address-range].
- R- Register Command: Used to display the contents of one or more registers. It also display the states of the flags.
  Syntax: G[=address]

- T - Trace Command : It is used to run a program in single - step mode
  Syntax : T = address (step).
- D - Display or Dump Command : It is used to display the contents of specified memory locations.
  Syntax : D address - range or D address.
- E - Enter Command : It is used to enter the data or machine code. Its default register is the DS.
  Syntax : E address
- F - Fill command : It is used to fill the specified range of locations with the values given in a list.
  Syntax : F address - range values.
- M - Move Command : It copies the block of data from one memory area to another.
  Syntax : M range address.
- S - Search Command : It is used to search the specified memory range for the specified list of bytes. The default register is the DS.
  Syntax : S range list.
- N - Save Command : It is used to save a file.
  Syntax : N filename.com.
- W - Write Command
- L - Load Command
- Q - Quit Command.

Assembler

- An assembler is a computer program for translating assembly language - essentially, a mnemonic representation of machine language - into object code.
- One-Pass Assembler: Which goes through an ALP only once and produces its object codes.
- Two-Pass Assembler: Which goes through an ALP only twice and produces its object codes.
- An ALP contains two types of statements: Instructions, Directives.
- Examples: MASM (microsoft ASsembler)

Directives:

- DB- Define Byte: It direct the assembler to reserve one byte of memory and initialize that byte with the specified value.
- DW - Define Word: It direct the assembler to reserve two bytes of memory and initialize those bytes with the specifed value.
- DD - Define Double Word: It defines a double word (4 bytes) type variable. It direct the assembler to reserve four bytes double coord of memory and initialize those bytes with the specified value.

- DQ - Define Quad Word: It defines a quad word (8 bytes) type variable. It direct the assembler to reserve eight byte of memory and initialize those bytes with the specified values.
- SEGMENT: It indicate the beginning of a logical segment.
  Syntax: Segment name SEGMENT [word/public]
- END: It indicate the end of the program.
- ASSUME: This tells the assembler the name of a logical segment, which is to be used for a specified segment.
- OFFSET: It determines the offset.

MASM (Microsoft Macro Assembler)

- The Microsoft Macro Assembler (abbreviated MASM) is an assembler for the x86 family of microprocessors. It was originally produced by Microsoft for development work on their MS-DOS operating system, and was for some time the most popular assembler available for that operating system.
- Steps to work on MASM:

* mount c c:\8086
* Save file with filename.asm
* C:\>masm>filename.asm
* C:\>link filename
* C:\>filename.

Result

Studied the assembler and debugging commands and CO_2 is attained.

19 / 12 / 2022

Experiment No: 10

PROGRAM TO ADD TWO 16-BIT NUMBERS

Aim

To write an assembly language program to add two 16 bit numbers.

Algorithm

Step 1: Start

Step 2: Initialize the data segment

Step 3: Read two numbers from user.

Step 4: Subtract 30 from each digit and continue then to form two numbers.

Step 5: Add the number.

Step 6: Add 30 to each digit, continue them and display the sum.

Step 7: Stop.

Result

Successfully implemented the assembly language Program to add two 16 bit numbers and output is verified . It satisfies CO_2.

```
//ADD TWO 16 BIT NUMBERS

ASSUME CS:CODE,DS:DATA
DATA SEGMENT
MSG1 DB 0AH, 'Enter the first number: $'
MSG2 DB 0AH, 'Enter the second number: $'
MSG3 DB 0AH, 'The sum is: $'
DATA ENDS
CODE SEGMENT
START:
MOV AX,DATA
MOV DS,AX
MOV DX,OFFSET MSG1
MOV AH,09H
INT 21H
MOV AH,01H
INT 21H
SUB AL,30H
MOV BH,0AH
MUL BH
MOV BH,AL
MOV AH,01H
INT 21H
SUB AL,30H
ADD BH,AL
MOV AH,01H
INT 21H
SUB AL,30H
MOV CH,0AH
MUL CH
MOV CH,AL
MOV AH,01H
INT 21H
SUB AL,30H
ADD CH,AL

MOV DX,OFFSET MSG2
MOV AH,09H
INT 21H
MOV AH,01H
INT 21H
SUB AL,30H
MOV BL,0AH
MUL BL
MOV BL,AL
MOV AH,01H
INT 21H
SUB AL,30H
ADD BL,AL
MOV AH,01H
```

Output

Enter the first number: 0100

Enter the second number: 0001

The sum is : 0101

07/01/2023

30

Experiment No: 11

PROGRAM TO REVERSE A STRING.

Aim

Write Assembly Language Program to accept a string and print it in reverse form.

Algorithm

Step 1: Start
Step 2: Read a string from the user.
Step 3: Traverse through the string
Step 4: Push the character in the stack.
Step 5: Count the number of characters
Step 6: Load the starting address of the string.
Step 7: Pop the top character of the stack until count is not equal to zero
Step 8: Put the character and reduce the count and reverse the address
Step 9: Continue until the count is greater than zero
Step 10: Load the effective address of string in dx using LEA command.
Step 11: Print the string by calling the intrupt with 09H in AH.
Step 12: The string must be terminated by '$' sign
Step 13: Stop.

Result

Successfully implemented the assembly language program to accept a string and print it in reverse form and output is verified. It satisfies CO2.

//REVERSE A STRING

```
.model small
.stack 100H
.data

MSG1 DB 13,10, "Enter String : $"
MSG2 DB 13,10, "Reverse String is : $"
.code

MOV AX, @DATA
MOV DS, AX
LEA DX, MSG1
MOV AH, 09H
INT 21H
MOV CX,00H
READ:
MOV AH, 01
INT 21H
CMP AL, 0DH
JE AHEAD
PUSH AX
INC CX
JMP READ
AHEAD:
LEA DX, MSG2
MOV AH,09H
INT 21H
DISPLAY:
MOV AH,02H
POP DX
INT 21H
LOOP DISPLAY
MOV AH,4CH
INT 21H
END
```

Output

Enter string : Hello

Reverse string is : olleh

09/01/2023

Experiment No: 12

INTERFACING WITH STEPPER MOTOR

Aim

Interfacing stepper motor using 8086 Trainer kit.

Program

Address	Mnemonics	Comments
1000	MOV DI, 2000	Move 2000 to DI
1004	MOV CL, 04	Move 04 to CL
1007	MOV AL, [DI]	Move [DI] to AL
1009	OUT C0, AL	Output port added to C0
100B	MOV BX, 1010	Move 1010 to BX
100F	DEC BX	Decrement BX
1010	JNZ 100F	Jump if not zero to 100F
1012	INC DI	Increment DI
1013	LOOP 1007	Loop to 1007
1015	JMP 1000	Jump to 1000
1018	HLT	Halt.

Result

Interfacing stepper motor with 8086 and the output is verified. It satisfies CO3.

Output

Clockwise

```
SB 2000
2000-09
2001-05
2002-06
2003-0A
2004-09
RESET
Go 1000
```

Anticlockwise

```
SB 2000
2000-09
2001-0A
2002-06
2003-05
2004-09
RESET
GO 1000
```

09/01/2023

Experiment No: 13

INTERFACING WITH 8255 [MODE 0]

Aim

To interface 8255 using 8086 trainer kit

Initialise port A as Input Port and port B as Output Port in mode 0

Program

Address	Mnemonics	Comments.
1000	MOV AL,90	Move 90 to AL
1003	OUT C6,AL	Output Port add to C6
1005	IN AL,C0	Input port
1007	OUT C2,AL	Output port add to C2
1009	MOV SI,1100	Move 1100 to SI
100D	MOV (SI),AL	Move AL to [SI]
100F	HLT	Stop program execution.

Result

The program to implement interfacing 8255 (mode 0) using 8086 trainer kit has been implemented and output is verified. It satisfies CO3.