Netaji Subhas University of Technology

Practical Report

Microprocessors and Microcontrollers

Kushagra Lakhwani

2021UCI8036

Computer Science Engineering (Internet of Things)

Semester 3

Department of Computer Science & Engineering

September 19, 2023

Contents

1 The Fibonacci sequence

1.1 Objective

Write an assembly program to generate the numbers of the Fibonacci series.

1.2 Implementation

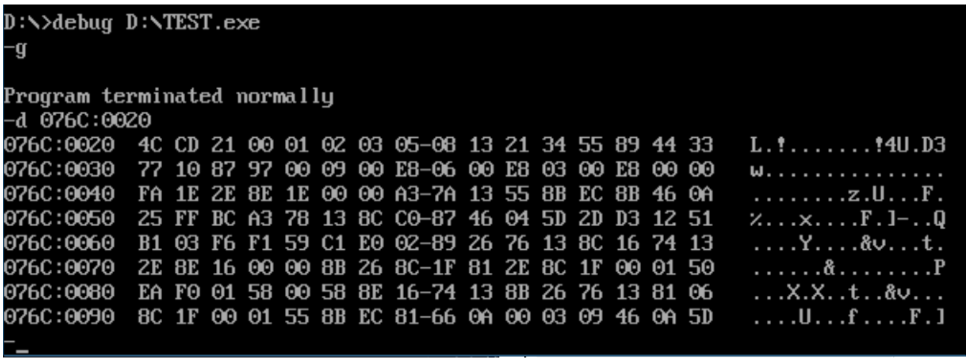
The Fibonacci sequence is defined as follows:

$$
\begin{aligned}
F_0 &= 0 \\
F_1 &= 1 \\
F_n &= F_{n-1} + F_{n-2}
\end{aligned}
$$

1.2.1 Assembly code

```
; calculate the fibonacci sequence

.MODEL SMALL
.DATA
    FIB  DB ?
    CNT  DB 10H    ; Initialize the counter for the no. of fibNo needed
.CODE
    START:
          MOV  AX,@DATA
          MOV  DS,AX
          LEA  DI, FIB
          MOV  CL,CNT
          MOV  AX,00H
          MOV  BX,01H
    L1:
          ADD  AX,BX
          DAA
          MOV  [DI],AX
          MOV  AX,BX
          MOV  BX,[DI]

          INC  DI
          LOOP L1
          MOV  AH,4CH

          INT  21H
    END START
CODE ENDS

```

1.3 Output



```
D:\>debug D:\TEST.exe
-g

Program terminated normally
-d 076C:0020
076C:0020  4C CD 21 00 01 02 03 05-08 13 21 34 55 89 44 33   L.!.......!4U.D3
076C:0030  77 10 87 97 00 09 00 E8-06 00 E8 03 00 E8 00 00   w...............
076C:0040  FA 1E 2E 8E 1E 00 00 A3-7A 13 55 8B EC 8B 46 0A   ........z.U...F.
076C:0050  25 FF BC A3 78 13 8C C0-87 46 04 5D 2D D3 12 51   %...x....F.]-..Q
076C:0060  B1 03 F6 F1 59 C1 E0 02-89 26 76 13 8C 16 74 13   ....Y....&v...t.
076C:0070  2E 8E 16 00 00 8B 26 8C-1F 81 2E 8C 1F 00 01 50   ......&........P
076C:0080  EA F0 01 58 00 58 8E 16-74 13 8B 26 76 13 81 06   ...X.X..t..&v...
076C:0090  8C 1F 00 01 55 8B EC 81-66 0A 00 03 09 46 0A 5D   ....U...f....F.]
-_
```

Figure 1: Fibonacci program

2 Artithmetic instructions

2.1 Objective

Write an assembly program to perform the following operations:

- Addition
- Subtraction
- Multiplication
- Division

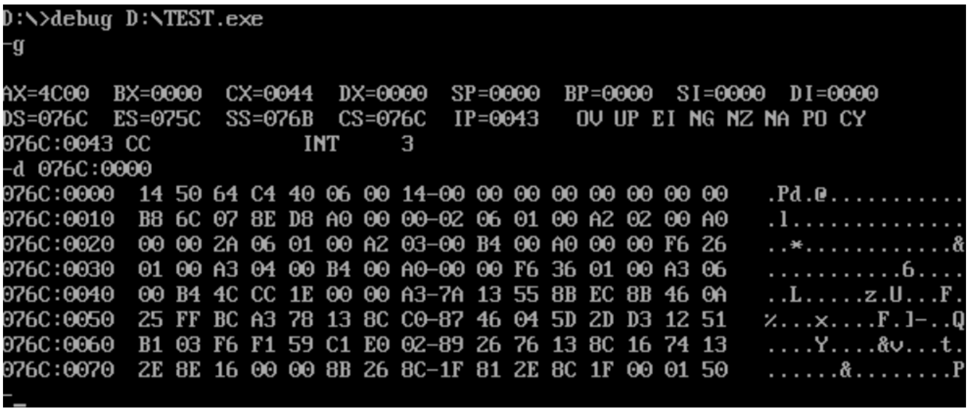
2.2 Implementation

Using the 8086 microprocessor's instruction set we can perform the above operations.

2.2.1 Assembly code

```
; 8086 assembly program to show the arithmetic instruction set
Data SEGMENT
      A          DB 14H
      B          DB 50H
      Sum        DB ?        ; word to store the sum of A + B
      Difference DB ?        ; word to store the difference A - B
      Product    DW ?        ; word to store the product A * B
      Division   DW ?        ; word to store the division A / B
Data ENDS

Code SEGMENT
            ASSUME CS: Code, DS: Data
      START:
            MOV    AX, Data
            MOV    DS, AX
      ; Addition
            MOV    AL, A
            ADD    AL, B
            MOV    Sum, AL
      ; Subtraction
            MOV    AL, A
            SUB    AL, B
            MOV    Difference, AL
      ; Multiplication
            MOV    AH, 0                   ; clear AH
            MOV    AL, A
            MUL    B
            MOV    Product, AX
      ; Division
            MOV    AH, 0                   ; clear AH
            MOV    AL, A
            DIV    B
            MOV    Division, AX
      ; Halt
            MOV    AH, 4CH
            INT    3H
Code ENDS

END
```

2.3 Output



```
D:\>debug D:\TEST.exe
-g

AX=4C00  BX=0000  CX=0044  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=076C  ES=075C  SS=076B  CS=076C  IP=0043   OV UP EI NG NZ NA PO CY
076C:0043 CC            INT     3
-d 076C:0000
076C:0000  14 50 64 C4 40 06 00 14-00 00 00 00 00 00 00 00   .Pd.@...........
076C:0010  B8 6C 07 8E D8 A0 00 00-02 06 01 00 A2 02 00 A0   .l..............
076C:0020  00 00 2A 06 01 00 A2 03-00 B4 00 A0 00 00 F6 26   ..*............&
076C:0030  01 00 A3 04 00 B4 00 A0-00 00 F6 36 01 00 A3 06   ...........6....
076C:0040  00 B4 4C CC 1E 00 00 A3-7A 13 55 8B EC 8B 46 0A   ..L.....z.U...F.
076C:0050  25 FF BC A3 78 13 8C C0-87 46 04 5D 2D D3 12 51   %...x....F.]-..Q
076C:0060  B1 03 F6 F1 59 C1 E0 02-89 26 76 13 8C 16 74 13   ....Y....&v...t.
076C:0070  2E 8E 16 00 00 8B 26 8C-1F 81 2E 8C 1F 00 01 50   ......&........P
-_
```

Figure 2: Arithmetic instructions

3 Sorting

3.1 Objective

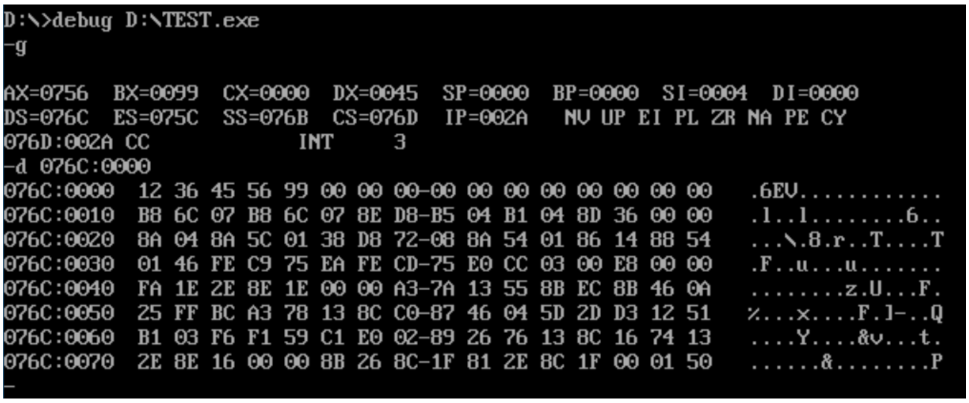
Write an assembly program to sort a list of numbers in ascending order.

3.2 Implementation

3.2.1 Assembly code

```
DATA SEGMENT
      STRING1 DB 99H, 12H, 56H, 45H, 36H
DATA ENDS

CODE SEGMENT
            ASSUME CS:CODE, DS:DATA
      START:
            MOV    AX, DATA

            MOV    AX, DATA
            MOV    DS, AX

            MOV    CH, 04H

      UP2:  MOV    CL, 04H
            LEA    SI, STRING1

      UP1:  MOV    AL, [SI]
            MOV    BL, [SI+1]
            CMP    AL, BL
            JC     DOWN
            MOV    DL, [SI+1]
            XCHG   [SI], DL
            MOV    [SI+1], DL

      DOWN: INC    SI
            DEC    CL
            JNZ    UP1
            DEC    CH
            JNZ    UP2

            INT    3
CODE ENDS
END START
```

3.3 Output



```
D:\>debug D:\TEST.exe
-g

AX=0756  BX=0099  CX=0000  DX=0045  SP=0000  BP=0000  SI=0004  DI=0000
DS=076C  ES=075C  SS=076B  CS=076D  IP=002A   NV UP EI PL ZR NA PE CY
076D:002A CC            INT     3
-d 076C:0000
076C:0000  12 36 45 56 99 00 00 00-00 00 00 00 00 00 00 00   .6EV............
076C:0010  B8 6C 07 B8 6C 07 8E D8-B5 04 B1 04 8D 36 00 00   .l..l........6..
076C:0020  8A 04 8A 5C 01 38 D8 72-08 8A 54 01 86 14 88 54   ...\.8.r..T....T
076C:0030  01 46 FE C9 75 EA FE CD-75 E0 CC 03 00 E8 00 00   .F..u...u.......
076C:0040  FA 1E 2E 8E 1E 00 00 A3-7A 13 55 8B EC 8B 46 0A   ........z.U...F.
076C:0050  25 FF BC A3 78 13 8C C0-87 46 04 5D 2D D3 12 51   %...x....F.]-..Q
076C:0060  B1 03 F6 F1 59 C1 E0 02-89 26 76 13 8C 16 74 13   ....Y....&v...t.
076C:0070  2E 8E 16 00 00 8B 26 8C-1F 81 2E 8C 1F 00 01 50   ......&........P
-
```

Figure 3: Sorting a list of numbers

4 Factorial

4.1 Objective

To write a program to calculate the factorial of a number.

4.2 Implementation

The factorial of a number n is calculated using the following formula:

$$n! = n \times (n-1) \times (n-2) \times \cdots \times 1 \tag{1}$$

4.2.1 Assembly code

```
; 8086 assembly program to calculate the factorial of a number

DATA SEGMENT
    N    DW 7h     ; factorial to calculate
DATA ENDS

CODE SEGMENT
          ASSUME CS:CODE, DS:DATA
    START:
          MOV    AX, DATA
          MOV    DS, AX
          MOV    AX, N
          MOV    BX, AX
          DEC    BX

    LOOP1:
          MUL    BX
          DEC    BX
          JNZ    LOOP1
          MOV    N, AX                ; store result in N
          INT    3h                   ; break to debugger
CODE ENDS
    END START
```

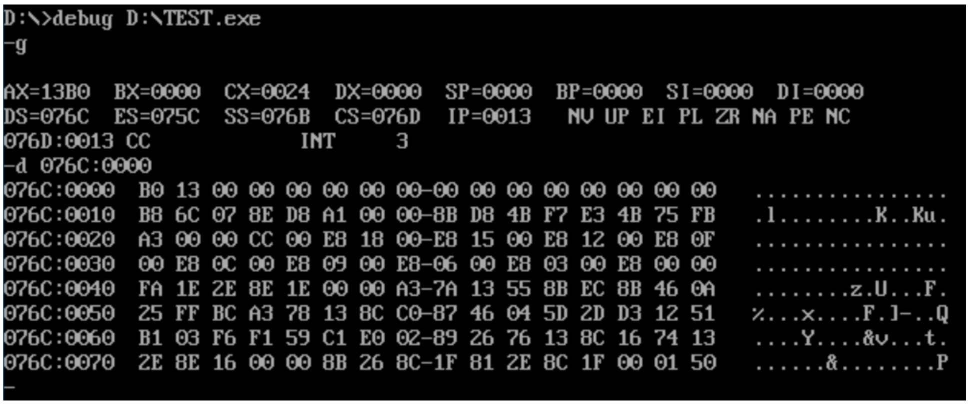
4.3 Output

4.3.1 Discussion

The factorial of 7:

$$7! = 5040_{10} = 13B0_{16} \tag{2}$$

Which can be seen in Figure 4 at the start of the DS segment.



```
D:\>debug D:\TEST.exe
-g

AX=13B0  BX=0000  CX=0024  DX=0000  SP=0000  BP=0000  SI=0000  DI=0000
DS=076C  ES=075C  SS=076B  CS=076D  IP=0013   NV UP EI PL ZR NA PE NC
076D:0013 CC            INT     3
-d 076C:0000
076C:0000  B0 13 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076C:0010  B8 6C 07 8E D8 A1 00 00-8B D8 4B F7 E3 4B 75 FB   .l........K..Ku.
076C:0020  A3 00 00 CC 00 E8 18 00-E8 15 00 E8 12 00 E8 0F   ................
076C:0030  00 E8 0C 00 E8 09 00 E8-06 00 E8 03 00 E8 00 00   ................
076C:0040  FA 1E 2E 8E 1E 00 00 A3-7A 13 55 8B EC 8B 46 0A   ........z.U...F.
076C:0050  25 FF BC A3 78 13 8C C0-87 46 04 5D 2D D3 12 51   %...x....F.]-..Q
076C:0060  B1 03 F6 F1 59 C1 E0 02-89 26 76 13 8C 16 74 13   ....Y....&v...t.
076C:0070  2E 8E 16 00 00 8B 26 8C-1F 81 2E 8C 1F 00 01 50   ......&........P
-
```

Figure 4: Factorial program

5 Square root

5.1 Objective

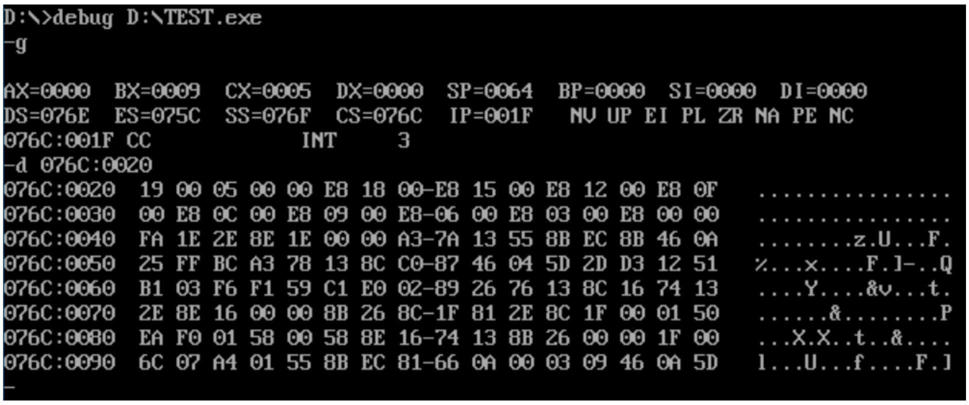
To write a program to calculate the square root of a number.

5.2 Assembly code

```
; 8086 assemply program to calculate the square root of a number

.MODEL SMALL
.STACK 100
.DATA                     ; Data segment starts
    NUM1 DW 0019H         ; Initialize num1 to 0019 (25 in decimal
    SQRT DW 01 DUP (?)    ; Reserve 1 word of uninitialised data space to offset sqrt

.CODE                      ; Code segment starts
    START:
          MOV AX, @DATA    ; Initialize data segment
          MOV DS, AX
          MOV AX, NUM1     ; Move the number(num1) to AX
          XOR BX, BX       ; XOR is performed and result is stored in BX
          MOV BX, 0001H    ; Initialize BX to 0001H
          MOV CX, 0001H    ; Initialize CX to 0001H
    LOOP1:SUB AX, BX       ; AX <- AX - BX
          JZ  LOOP2        ;  If zero flag is zero jump to loop2
          INC CX           ;  Increment CX by 1
          ADD BX, 0002H    ; BX <- BX + 0002H
          JMP LOOP1        ;  Jump to loop1
          INC CX           ;  Increment CX by 1
    LOOP2:MOV SQRT, CX     ;  Store result
          INT 03H          ;  halt to debugger

END START
```

Inspiration taken from [1].

5.3 Output



```
D:\>debug D:\TEST.exe
-g

AX=0000  BX=0009  CX=0005  DX=0000  SP=0064  BP=0000  SI=0000  DI=0000
DS=076E  ES=075C  SS=076F  CS=076C  IP=001F   NV UP EI PL ZR NA PE NC
076C:001F CC            INT     3
-d 076C:0020
076C:0020  19 00 05 00 00 E8 18 00-E8 15 00 E8 12 00 E8 0F   ................
076C:0030  00 E8 0C 00 E8 09 00 E8-06 00 E8 03 00 E8 00 00   ................
076C:0040  FA 1E 2E 8E 1E 00 00 A3-7A 13 55 8B EC 8B 46 0A   ........z.U...F.
076C:0050  25 FF BC A3 78 13 8C C0-87 46 04 5D 2D D3 12 51   %...x....F.]-..Q
076C:0060  B1 03 F6 F1 59 C1 E0 02-89 26 76 13 8C 16 74 13   ....Y....&v...t.
076C:0070  2E 8E 16 00 00 8B 26 8C-1F 81 2E 8C 1F 00 01 50   ......&........P
076C:0080  EA F0 01 58 00 58 8E 16-74 13 8B 26 00 00 1F 00   ...X.X..t..&....
076C:0090  6C 07 A4 01 55 8B EC 81-66 0A 00 03 09 46 0A 5D   l...U...f....F.]
-
```

Figure 5: Square root program

6 Move data

6.1 Objective

Move data from one memory location to another.

6.2 Assembly code

```
; 8086 assembly program to transfer 10 bytes
; from 2000:0000 to 3000:0000

Code SEGMENT
        ASSUME CS: Code
        MOV    AX, 2000H
        MOV    DS, AX
        MOV    AX, 3000H
        MOV    ES, AX
        MOV    SI, 0000H
        MOV    DI, 0000H
        MOV    CX, 000AH
        CLD
        REP    MOVSB
        INT    3
Code ENDS
END
```

7 The 8259 Interface Chip

7.1 Background

8259 microprocessor is defined as *Programmable Interrupt Controller (PIC)* microprocessor. There are 5 hardware interrupts and 2 hardware interrupts in 8085 and 8086 respectively. But by connecting 8259 with CPU, we can increase the Interrupt handling capability. 8259 combines the multi-interrupt input sources into a single interrupt output. Interfacing of single PIC provides 8 interrupts inputs from *IR0-IR7.*

For example, interfacing of 8085 and 8259 increases the interrupt handling capability of 8085 microprocessor from 5 to 8 interrupt levels.

7.2 Features of 8259

- Intel 8259 is designed for Intel 8085 and Intel 8086 microprocessor.
- It can be programmed either in level triggered or in edge triggered interrupt level.
- We can mask individual bits of interrupt request register.
- We can increase interrupt handling capability up to 64 interrupt level by cascading further 8259 PIC.
- Clock cycle is not required
- It can be programmed in 8085 and 8086 microprocessor.

7.3 Pin Description

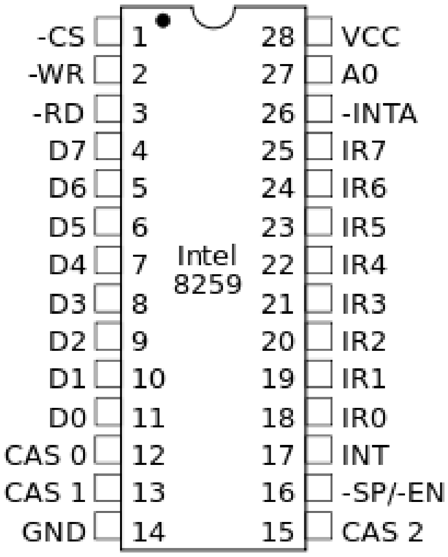


Figure 6: Pin Diagram of 8259

7.4 Block Diagram

The block diagram consists of 8 blocks which are:

- Data bus buffer

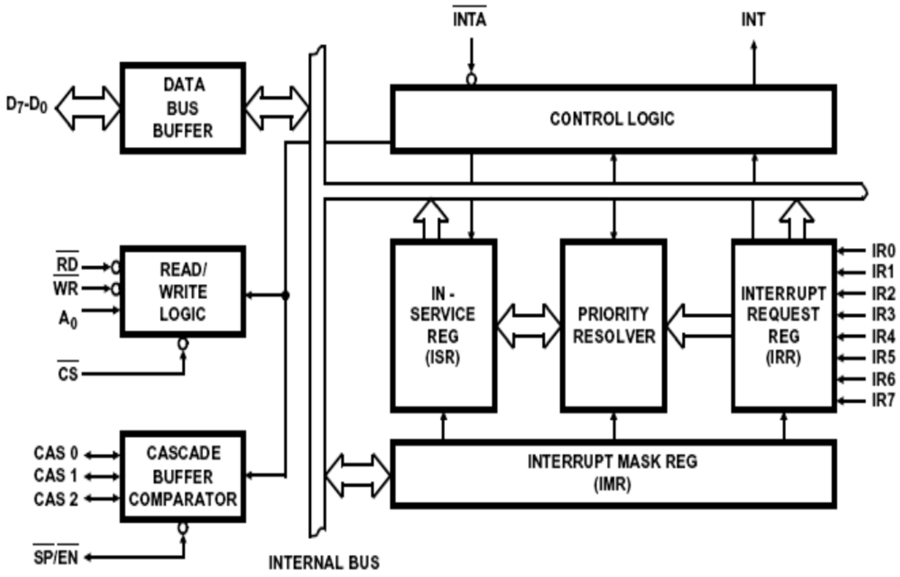


Figure 7: Block Diagram of 8259

- Read/Write Logic
- Cascade Buffer Comparator
- Control Logic
- Priority Resolve
- Interrupt Request Register (IRR)
- Interrupt Service Register (ISR)

List of Figures

References

[1] jntuimplab, “8086 programs blog post.” `https://jntuimplab.blogspot.com/2008/01/experiment-6.html`.