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

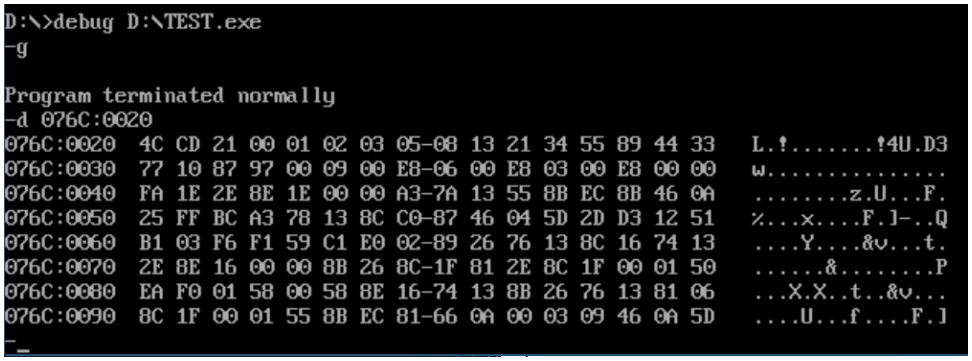

Figure 1: Fibonacci program

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

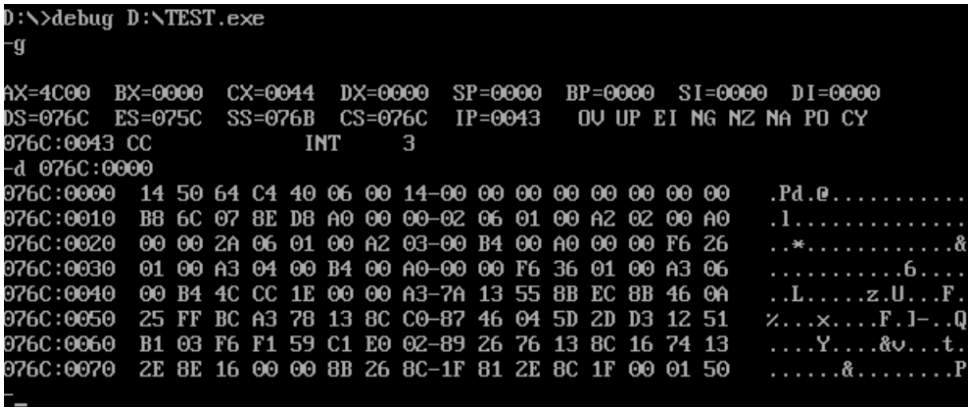

Figure 2: Arithmetic instructions

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

```
1  DATA SEGMENT
2        STRING1 DB 99H, 12H, 56H, 45H, 36H
3  DATA ENDS
4
5  CODE SEGMENT
6              ASSUME CS:CODE, DS:DATA
7        START:
8              MOV    AX, DATA
9
10             MOV    AX, DATA
11             MOV    DS, AX
12
13             MOV    CH, 04H
14
15       UP2:  MOV    CL, 04H
16             LEA    SI, STRING1
17
18       UP1:  MOV    AL, [SI]
19             MOV    BL, [SI+1]
20             CMP    AL, BL
21             JC     DOWN
22             MOV    DL, [SI+1]
23             XCHG   [SI], DL
24             MOV    [SI+1], DL
25
26       DOWN: INC    SI
27             DEC    CL
28             JNZ    UP1
29             DEC    CH
30             JNZ    UP2
31
32             INT    3
33 CODE ENDS
34 END START
```

3.3 Output

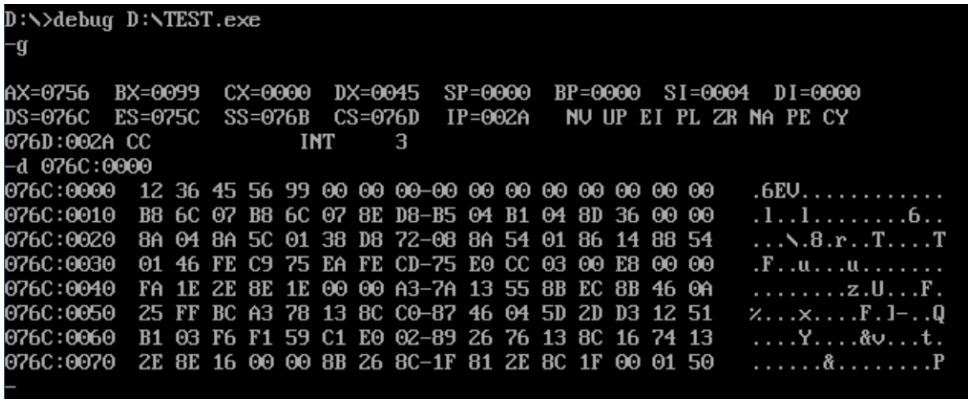

Figure 3: Sorting a list of numbers

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.

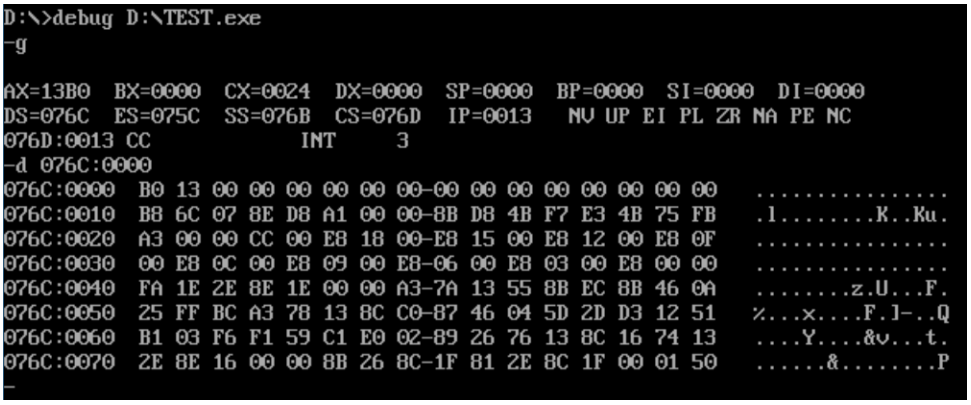

Figure 4: Factorial program

5 Square root

5.1 Objective

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

5.1.1 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].

5.2 Output

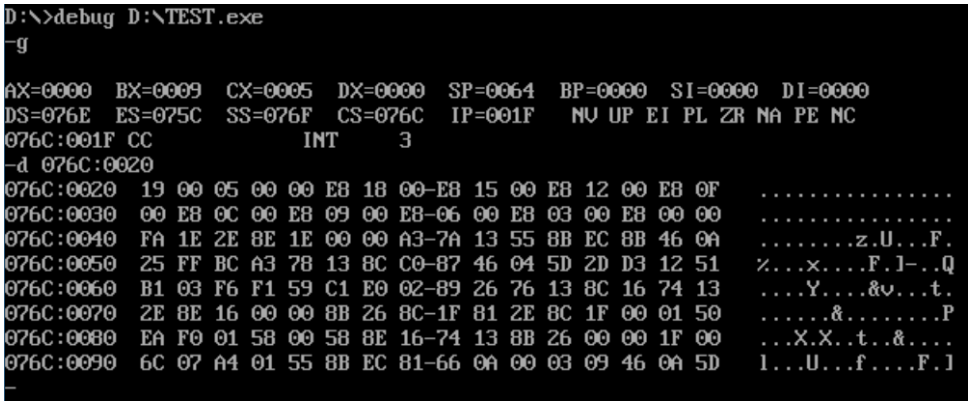

Figure 5: Square root program

6 Move data

6.1 Objective

Move data from one memory location to another.

6.1.1 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

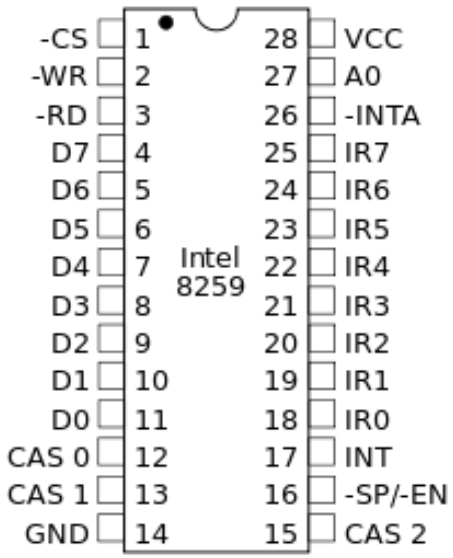

Figure 6: Pin Diagram of 8259

7.4 Block Diagram

The block diagram consists of 8 blocks which are:

- Data bus buffer

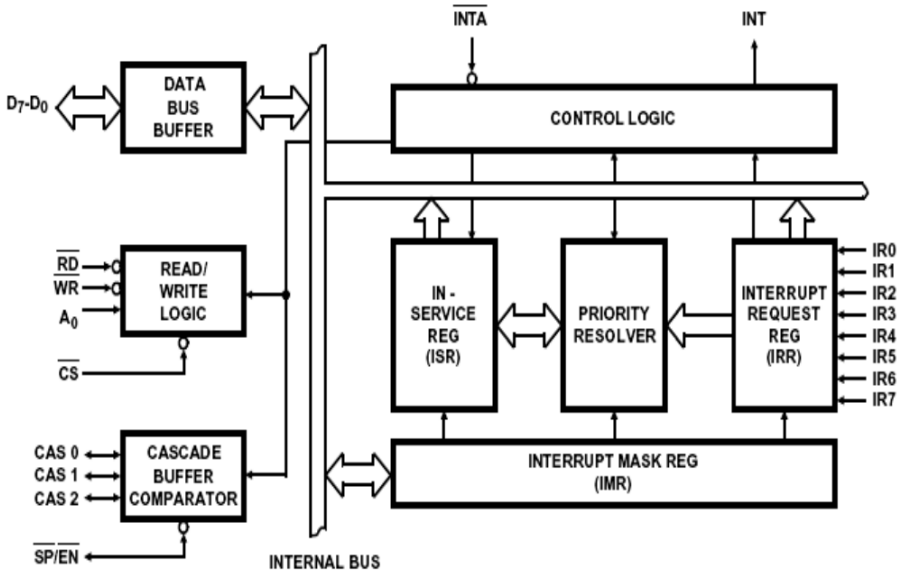

Figure 7: Block Diagram of 8259

- 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`.