



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 |
| 1.1 | Objective | 1 |
| 1.2 | Implementation | 1 |
| 1.2.1 | Assembly code | 1 |
| 1.3 | Output | 2 |
| 2 | Arithmetic instructions | 3 |
| 2.1 | Objective | 3 |
| 2.2 | Implementation | 3 |
| 2.2.1 | Assembly code | 3 |
| 2.3 | Output | 4 |
| 3 | Sorting | 5 |
| 3.1 | Objective | 5 |
| 3.2 | Implementation | 5 |
| 3.2.1 | Assembly code | 5 |
| 3.3 | Output | 6 |
| 4 | Factorial | 7 |
| 4.1 | Objective | 7 |
| 4.2 | Implementation | 7 |
| 4.2.1 | Assembly code | 7 |
| 4.3 | Output | 8 |
| 4.3.1 | Discussion | 8 |
| 5 | Square root | 9 |
| 5.1 | Objective | 9 |
| 5.1.1 | Assembly code | 9 |
| 5.2 | Output | 10 |
| 6 | Move data | 11 |
| 6.1 | Objective | 11 |
| 6.1.1 | Assembly code | 11 |
| 7 | The 8259 Interface Chip | 12 |
| 7.1 | Background | 12 |
| 7.2 | Features of 8259 | 12 |
| 7.3 | Pin Description | 12 |
| 7.4 | Block Diagram | 12 |

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:

$$F_0 = 0$$

$$F_1 = 1$$

$$F_n = F_{n-1} + F_{n-2}$$

1.2.1 Assembly code

```
1  ; calculate the fibonacci sequence
2
3  .MODEL SMALL
4  .DATA
5      FIB DB ?
6      CNT DB 10H      ; Initialize the counter for the no. of fibNo needed
7  .CODE
8      START:
9          MOV AX,@DATA
10         MOV DS,AX
11         LEA DI, FIB
12         MOV CL,CNT
13         MOV AX,00H
14         MOV BX,01H
15     L1:
16         ADD AX,BX
17         DAA
18         MOV [DI],AX
19         MOV AX,BX
20         MOV BX,[DI]
21
22         INC DI
23         LOOP L1
24         MOV AH,4CH
25
26         INT 21H
27     END START
28 CODE ENDS
29
```

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.l-..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.l
_
```

Figure 1: Fibonacci program

2 Arithmetic 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

```
1 ; 8086 assembly program to show the arithmetic instruction set
2 Data SEGMENT
3     A          DB 14H
4     B          DB 50H
5     Sum        DB ?           ; word to store the sum of A + B
6     Difference DB ?           ; word to store the difference A - B
7     Product    DW ?           ; word to store the product A * B
8     Division   DW ?           ; word to store the division A / B
9 Data ENDS
10
11 Code SEGMENT
12     ASSUME CS: Code, DS: Data
13     START:
14         MOV     AX, Data
15         MOV     DS, AX
16     ; Addition
17         MOV     AL, A
18         ADD     AL, B
19         MOV     Sum, AL
20     ; Subtraction
21         MOV     AL, A
22         SUB     AL, B
23         MOV     Difference, AL
24     ; Multiplication
25         MOV     AH, 0           ; clear AH
26         MOV     AL, A
27         MUL     B
28         MOV     Product, AX
29     ; Division
30         MOV     AH, 0           ; clear AH
31         MOV     AL, A
32         DIV     B
33         MOV     Division, AX
34     ; Halt
35         MOV     AH, 4CH
36         INT     3H
37 Code ENDS
38
39 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.l-..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

```
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

```
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 NU 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 .6EU.....
076C:0010 B8 6C 07 B8 6C 07 8E D8-B5 04 B1 04 8D 36 00 00 .1..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.l-..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:

$$n! = n \times (n - 1) \times (n - 2) \times \cdots \times 1 \quad (1)$$

4.2.1 Assembly code

```
1  ; 8086 assembly program to calculate the factorial of a number
2
3  DATA SEGMENT
4      N      DW 7h      ; factorial to calculate
5  DATA ENDS
6
7  CODE SEGMENT
8      ASSUME CS:CODE, DS:DATA
9      START:
10         MOV     AX, DATA
11         MOV     DS, AX
12         MOV     AX, N
13         MOV     BX, AX
14         DEC     BX
15
16     LOOP1:
17         MUL     BX
18         DEC     BX
19         JNZ     LOOP1
20         MOV     N, AX      ; store result in N
21         INT     3h        ; break to debugger
22 CODE ENDS
23 END START
```

4.3 Output

4.3.1 Discussion

The factorial of 7:

$$7! = 5040_{10} = 13B0_{16} \quad (2)$$

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  NU 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.l-..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.

5.1.1 Assembly code

```
1 ; 8086 assembly program to calculate the square root of a number
2
3 .MODEL SMALL
4 .STACK 100
5 .DATA ; Data segment starts
6     NUM1 DW 0019H ; Initialize num1 to 0019 (25 in decimal)
7     SQRT DW 01 DUP (?) ; Reserve 1 word of uninitialised data space to offset
8     ↪ sqrt
9
10 .CODE ; Code segment starts
11     START:
12         MOV AX, @DATA ; Initialize data segment
13         MOV DS, AX
14         MOV AX, NUM1 ; Move the number(num1) to AX
15         XOR BX, BX ; XOR is performed and result is stored in BX
16         MOV BX, 0001H ; Initialize BX to 0001H
17         MOV CX, 0001H ; Initialize CX to 0001H
18     LOOP1: SUB AX, BX ; AX <- AX - BX
19             JZ LOOP2 ; If zero flag is zero jump to loop2
20             INC CX ; Increment CX by 1
21             ADD BX, 0002H ; BX <- BX + 0002H
22             JMP LOOP1 ; Jump to loop1
23             INC CX ; Increment CX by 1
24     LOOP2: MOV SQRT, CX ; Store result
25             INT 03H ; halt to debugger
26
27 END START
```

Inspiration taken from [1].

5.2 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  NU 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.l...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  1...U...f....F.l

```

Figure 5: Square root program

6 Move data

6.1 Objective

Move data from one memory location to another.

6.1.1 Assembly code

```
1  ; 8086 assembly program to transfer 10 bytes
2  ; from 2000:0000 to 3000:0000
3
4  Code SEGMENT
5      ASSUME CS: Code
6      MOV     AX, 2000H
7      MOV     DS, AX
8      MOV     AX, 3000H
9      MOV     ES, AX
10     MOV     SI, 0000H
11     MOV     DI, 0000H
12     MOV     CX, 000AH
13     CLD
14     REP     MOVSB
15     INT     3
16 Code ENDS
17 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

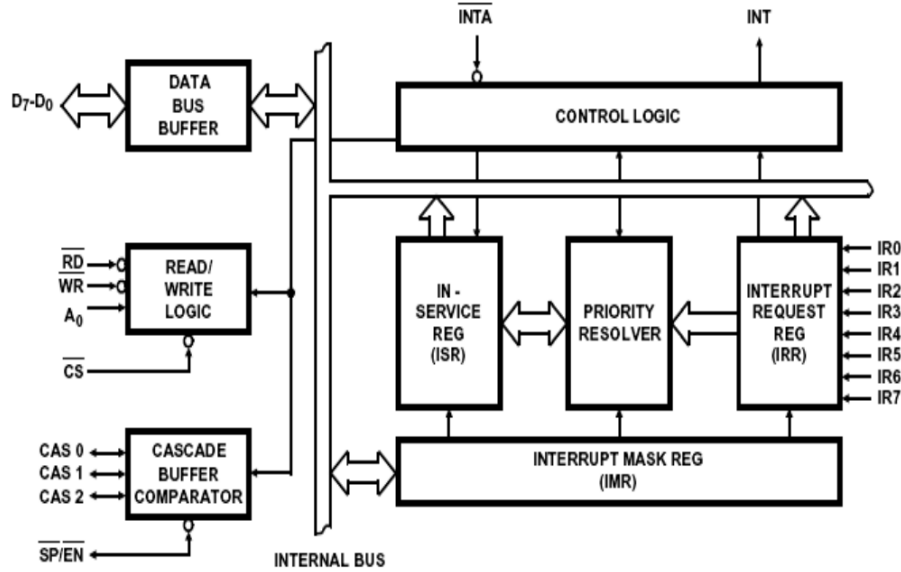


Figure 7: Block Diagram of 8259

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

List of Figures

| | | |
|---|-------------------------------------|----|
| 1 | Fibonacci program | 2 |
| 2 | Arithmetic instructions | 4 |
| 3 | Sorting a list of numbers | 6 |
| 4 | Factorial program | 8 |
| 5 | Square root program | 10 |
| 6 | Pin Diagram of 8259 | 12 |
| 7 | Block Diagram of 8259 | 13 |

References

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