Aravind Kannan Rathinasabapathi
195001022
Date: 06-08-2021
Ex. No. 4

Exercise 4 – Code Conversion
4A - Converting Hexadecimal to BCD

Aim:

Convert BCD to Hexadecimal.

Procedure for executing MASM:

1. Mount the local folder in the DOS-BOX using a temp disk name: `mount <disk-name> <folder-location>`
2. Change directory into the mounted disk: `<disk-name>: `
3. Assemble the instructions: `masm <file-name>.asm`
4. Link the object file(s) to produce an executable file(.exe): `link <file-name>.obj; ` Note that removal of semi-colon will make linking process interactive.
5. Debug the executable file to read the memory map and execute the program: `debug <file-name>.exe`. After entering debug mode,
   a. `d <segment:offset> ` - dump(read) memory map from the given location
   b. `e <segment:offset> ` - edit memory values from the given location. Use 'White space' to continue editing and 'new line' to exit editing.
   c. `u ` - unassemble code (with or without <segment:offset>)
   d. `g ` - execute the program
   e. `? ` - display command list
   f. `q` - quit the debugger

Algorithm:

1. Initialise data and extra segment using their respective registers.
2. Load the base address of data segment(ds) and extra segment(es) using an intermediate accumulator register(ax) as direct memory transfer is not allowed in 8086.
3. Load the AH with 00H and AL with given input, load BL with 10H, and perform division, to produce the quotient (BCD 1) at AL and remainder (BCD 0) at AH
4. Move the remainder (BCD 0) from AH to CL.
5. To give place value to the quotient (BCD 1) at AL, we multiply with 0AH by loading into BL.
6. We add the BCD 1 with place value at AL with BCD 0 at CL, to produce the hexadecimal value at AL.
7. Move AL to output.
8. Terminate the program.

Aravind Kannan Rathinasabapathi
195001022

Date: 06-08-2021
Ex. No. 4

Program:

Program	Comment
`; 4a: bcd2hex Mathematically`	Comment after ';'
`assume cs: code, ds: data`	Map CS to code segment, DS to data segment

`data segment`
`  input db 12H`
`  output db 00H`
`data ends`
	Initialise data segment and extra segment
	db = define a byte
	Initialise input, output

`code segment`
`start:  mov ax, data`
`  mov ds, ax`
	Initialise code segment
	Move the starting address of data segment in ax, then move ax to ds.

`  mov ah, 00H`
`  mov al, input`
`  mov bl, 10H`
`  div bl`
`  mov cl, ah`
	Load AL with input, and clear AH and load BL with 10H, to perform 8-bit division.
	AX / BL = (AL=quotient, AH=remainder)
	
	Move remainder (BCD 0) at AH to CL

`  mov bl, 0aH` `  mul bl`	Load BL with 0aH Multiply the quotient (BCD 1) at AL with 0aH to shift: AX = AL x BL
`  add al, cl`	Add AL(Shifted BCD 1) and CL (BCD 0)
`  mov output, al`	Move final result from AL to output

`  mov ah, 4cH`
`  int 21H`
`code ends`
`end start`
	Set ah = 4cH
	Call interrupt routine 21H for DOS, which terminates if ah = 4cH

Aravind Kannan Rathinasabapathi
195001022

Date: 06-08-2021
Ex. No. 4

Unassembled code:

```
-u
076B:0000 B86A07          MOV     AX,076A
076B:0003 8ED8            MOV     DS,AX
076B:0005 B400            MOV     AH,00
076B:0007 A00000          MOV     AL,[0000]
076B:000A B310            MOV     BL,10
076B:000C F6F3            DIV     BL
076B:000E 8ACC            MOV     CL,AH
076B:0010 B30A            MOV     BL,0A
076B:0012 F6E3            MUL     BL
076B:0014 02C1            ADD     AL,CL
076B:0016 A20100          MOV     [0001],AL
076B:0019 B44C            MOV     AH,4C
076B:001B CD21            INT     21
```

Snapshot of sample input and output:

Before execution:

```
-d 076a:0000
076A:0000  12 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0010  B8 6A 07 8E D8 B4 00 A0-00 00 B3 10 F6 F3 8A CC   .j..............
076A:0020  B3 0A F6 E3 02 C1 A2 01-00 B4 4C CD 21 1E 8A 5E   ..........L.!..^
076A:0030  F9 B7 00 D1 E3 8B 87 AE-16 3B 46 FE 77 09 89 46   .........;F.w..F
076A:0040  FE 8A 46 F9 88 46 F8 FE-46 F9 EB C9 8A 5E F8 B7   ..F..F..F....^..
076A:0050  00 8A 87 48 2F D0 D8 73-17 E8 B6 00 8A 5E F8 B7   ...H/..s.....^..
076A:0060  00 8A 87 48 2F D0 D8 73-07 53 B0 01 50 E8 73 01   ...H/..s.S..P.s.
076A:0070  A0 B6 2C 3A 46 F8 74 7E-C7 46 FA 00 00 8A 46 F8   ..,:F.t~.F....F.
```

After execution:

```
-g

Program terminated normally
-d 076a:0000
076A:0000  12 0C 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0010  B8 6A 07 8E D8 B4 00 A0-00 00 B3 10 F6 F3 8A CC   .j..............
076A:0020  B3 0A F6 E3 02 C1 A2 01-00 B4 4C CD 21 1E 8A 5E   ..........L.!..^
076A:0030  F9 B7 00 D1 E3 8B 87 AE-16 3B 46 FE 77 09 89 46   .........;F.w..F
076A:0040  FE 8A 46 F9 88 46 F8 FE-46 F9 EB C9 8A 5E F8 B7   ..F..F..F....^..
076A:0050  00 8A 87 48 2F D0 D8 73-17 E8 B6 00 8A 5E F8 B7   ...H/..s.....^..
076A:0060  00 8A 87 48 2F D0 D8 73-07 53 B0 01 50 E8 73 01   ...H/..s.S..P.s.
076A:0070  A0 B6 2C 3A 46 F8 74 7E-C7 46 FA 00 00 8A 46 F8   ..,:F.t~.F....F.
```

Aravind Kannan Rathinasabapathi
195001022

Date: 06-08-2021
Ex. No. 4

Result:

Program to convert BCD to Hexadecimal is assembled, executed and verified.

Aravind Kannan Rathinasabapathi
195001022

Date: 06-08-2021
Ex. No. 4

4B – Converting Hexadecimal to BCD

Aim:

Convert Hexadecimal to BCD.

Algorithm:

1. Initialise data and extra segment using their respective registers.
2. Load the base address of data segment(ds) and extra segment(es) using an intermediate accumulator register(ax) as direct memory transfer is not allowed in 8086.
3. Load the AH with 00H and AL with given input, load BL with 64H, and perform division, to produce the quotient (BCD 2) at AL and remainder at AH.
4. Move the quotient (BCD 2) at AL to msb.
5. Move AH to AL and clear AH, load BL with 0aH, and perform division, to produce the quotient (BCD 1) at AL and remainder (BCD 0) at AH.
6. To give unpack BCD we rotate AL (BCD 1) left by 4 bits, we load 04H to CL to implement the Rotate Left instruction.
7. We add the BCD 1 at AL with BCD 0 at AH, to produce the unpacked BCD values at AL.
8. Move AL to LSB.
9. Terminate the program.

Aravind Kannan Rathinasabapathi
195001022

Date: 06-08-2021
Ex. No. 4

Program:

Program	Comment
`; Prac - 4b: hex2bcd Mathematically`	Comment after ';'
`assume cs: code, ds: data`	Map CS to code segment, DS to data segment

`data segment`
`  input db 0FFH`
`  msb db 00H`
`  lsb db 00H`
`data ends`
	Initialise data segment and extra segment
	db = define a byte
	Initialise input, msb, lsb

`code segment`
`start:  mov ax, data`
`  mov ds, ax`
	Initialise code segment
	Move the starting address of data segment in ax, then move ax to ds.

`  mov ah, 00H`
`  mov al, input`
`  mov bl, 64H`
`  div bl`
	Load AL with input, and clear AH and load BL with 64H, to perform 8-bit division.
	AX / BL = (AL=quotient, AH=remainder)
	AL = BCD 2

`  mov msb, al`	Move quotient at AL to MSB

`  mov al, ah`
`  mov ah, 00H`
`  mov bl, 0aH`
`  div bl`
	Load AL with AH, and clear AH and load BL with 0aH, to perform 8-bit division.
	AX / BL = (AL=quotient, AH=remainder)
	AL = BCD 1; AH = BCD 0

`  mov cl, 04h` `  rol al, cl`	To give place value in base of BCD, we rotate the BCD 1 at AL to left by 4 bits. For this, we load cl, with 04H
`  add al, ah`	Add the place valued AL with AH
`  mov lsb, al`	Move the packed BCD to lsb

`  mov ah, 4cH`
`  int 21H`
`code ends`
`end start`
	Set ah = 4cH
	Call interrupt routine 21H for DOS, which terminates if ah = 4cH

Aravind Kannan Rathinasabapathi
195001022

Date: 06-08-2021
Ex. No. 4

Unassembled code:

```
D:\>debug M-H2B.EXE
-u
076B:0000 B86A07        MOV     AX,076A
076B:0003 8ED8          MOV     DS,AX
076B:0005 B400          MOV     AH,00
076B:0007 A00000        MOV     AL,[0000]
076B:000A B364          MOV     BL,64
076B:000C F6F3          DIV     BL
076B:000E A20100        MOV     [0001],AL
076B:0011 8AC4          MOV     AL,AH
076B:0013 B400          MOV     AH,00
076B:0015 B30A          MOV     BL,0A
076B:0017 F6F3          DIV     BL
076B:0019 B104          MOV     CL,04
076B:001B D2C0          ROL     AL,CL
076B:001D 02C4          ADD     AL,AH
076B:001F A20200        MOV     [0002],AL
076B:0022 B44C          MOV     AH,4C
076B:0024 CD21          INT     21
```

Snapshot of sample input and output:

Before execution:

```
-d 076a:0000
076A:0000  FF 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
076A:0010  B8 6A 07 8E D8 B4 00 A0-00 00 B3 64 F6 F3 A2 01   .j.........d....
076A:0020  00 8A C4 B4 00 B3 0A F6-F3 B1 04 D2 C0 02 C4 A2   ................
076A:0030  02 00 B4 4C CD 21 87 AE-16 3B 46 FE 77 09 89 46   ...L.!...;F.w..F
076A:0040  FE 8A 46 F9 88 46 F8 FE-46 F9 EB C9 8A 5E F8 B7   ..F..F..F....^..
076A:0050  00 8A 87 48 2F D0 D8 73-17 E8 B6 00 8A 5E F8 B7   ...H/..s.....^..
076A:0060  00 8A 87 48 2F D0 D8 73-07 53 B0 01 50 E8 73 01   ...H/..s.S..P.s.
076A:0070  A0 B6 2C 3A 46 F8 74 7E-C7 46 FA 00 00 8A 46 F8   ..,:F.t~.F....F.
```

After execution:

```
-g

Program terminated normally
-d 076a:0000
076A:0000  FF 02 55 00 00 00 00 00-00 00 00 00 00 00 00 00   ..U.............
076A:0010  B8 6A 07 8E D8 B4 00 A0-00 00 B3 64 F6 F3 A2 01   .j.........d....
076A:0020  00 8A C4 B4 00 B3 0A F6-F3 B1 04 D2 C0 02 C4 A2   ................
076A:0030  02 00 B4 4C CD 21 87 AE-16 3B 46 FE 77 09 89 46   ...L.!...;F.w..F
076A:0040  FE 8A 46 F9 88 46 F8 FE-46 F9 EB C9 8A 5E F8 B7   ..F..F..F....^..
076A:0050  00 8A 87 48 2F D0 D8 73-17 E8 B6 00 8A 5E F8 B7   ...H/..s.....^..
076A:0060  00 8A 87 48 2F D0 D8 73-07 53 B0 01 50 E8 73 01   ...H/..s.S..P.s.
076A:0070  A0 B6 2C 3A 46 F8 74 7E-C7 46 FA 00 00 8A 46 F8   ..,:F.t~.F....F.
```

Result:

Program to convert Hexadecimal to BCD is assembled, executed and verified.