CS422
COMPUTER PERIPHERALS AND INTERFACES

Lab Report

Assignment 1

Assembly Programs for 16-bit Addition, Subtraction, Multiplication and Division on 8085 Microprocessor

Group 11

1. Rajat Khanduja (09010137)
2. Raman Anurag (09010139)
3. Rovin Bhandari (09010144)
4. Pranav Gupta (09010161)

9th August, 2012

Contents

Programs

Following are the programs for the addition, subtraction, multiplication & division operations.

1. 16-Bit Addition

This program adds the two numbers located at addresses 8800-01 and 8802-03 and saves the sum at address 8804-05. The overflow word is saved at 8806-07. It assumes lower byte to be stored at lower address in the address pair. So, the lower byte of sum is stored at 8806 and higher byte at 8807.
Start Address: A400
Uses: Registers A, B/C, D/E, F, H/L
Program:

```
cpu "8085.tbl"
hof "int8"

org 9000h

LHLD 9050h ; load first number from memory to the HL
pair.
XCHG ; load HL into DE pair.
LHLD 9052h ; load second number from memory to the HL
pair.
MVI C,00h ; initialize C to 0
DAD D ; add the DE pair to the HL pair with carry flag
set (if there is a carry at all).
JNC STORERESULT
INR C ; takes care of the carry.
STORERESULT: SHLD 9054h ; move HL pair to memory.
MOV A,C ; move carry to A from C.
STA 9056h ; store A into memory.
RST 5
```

2. 16-Bit Subtraction

This program subtracts the number located at address 8802-03 from the number at 8800-01 and saves the result at address 8804-05. It assumes lower byte to be stored at lower address in the address pair. So, the lower byte of result is stored at 8806 and higher byte at 8807.
Start Address: A500
Uses: Registers A, B/C, F, H/L
Program:

```
cpu "8085.tbl"
hof "int8"

org 9000h

LHLD 9050h ; Load from C050
XCHG       ; DE <- HL (rather, ED <- LH)
LHLD 9052h ; Load from C052
MOV A,L    ; A <- L
SUB E      ; A = A - E
MOV L,A    ; L <- A
```

```
MOV A,H    ; A <- H
SBB D      ; Subtract with borrow.
MOV H,A    ; H <- A
STORERESULT:  SHLD  9054h  ;  Store  result  back  to  the
memory
RST 05
```

3. 16-Bit Multiplication

This program multiplies the two numbers located at addresses 8800-01 and 8802-03 and saves the product at address 8804-05. The overflow word is neglected in this case because of insufficient internal registers. It assumes lower byte to be stored at lower address in the address pair. So, the lower byte of product is stored at 8806 and higher byte at 8807.
Start Address: A600
Uses: Registers A, B/C, D/E, F, H/L
Program:

```
cpu "8085.tbl"
hof "int8"

org 9000h

LXI  H,9050h  ;  load  address  of  lower  8  bits  of
multiplier into H.
MOV C,M ; load lower 8 bits of multiplier into C.
LXI  H,9051h  ;  load  address  of  upper  8  bits  of
multiplier into H.
MOV B,M ; load upper 8 bits of multiplier into B.
LXI  H,9052h  ;  load  address  of  lower  8  bits  of
multiplicand into H.
MOV E,M ; load lower 8 bits of multiplier into E.
LXI  H,9053h  ;  load  address  of  upper  8  bits  of
multiplicand into H.
MOV D,M ; load upper 8 bits of multiplier into D.
MVI L,00H ; initialize L to 0
MVI H,00H ; initialize H to 0
; repeated addition
loop: MVI A,00H
ORA B ; copy B to A
JNZ decr ; if B is non-zero, goto decr
ORA C ; copy C to A
JZ exit ; exit if C is 0
decr: DAD D ; add DE pair to HL pair.
DCX B ; decrement BC pair by 1
JMP loop
STORERESULT: SHLD 9054h
RST 5
```

4. 16-Bit Division

This program divides the number located at address 8800-01 from the number at 8802-03 and saves the result at address 8804-05. Integer division is performed, i.e. the remainder is neglected. It assumes lower byte to be stored at lower address in the address pair. So, the lower byte of result is stored at 8806 and higher byte at 8807.
Start Address: A700
Uses: Registers A, B/C, D/E, F, H/L

Program:

```
cpu "8085.tbl"
hof "int8"

org 9000h

LHLD 9050h    ; Load divisor from memory location C050
to HL pair.
XCHG        ; Transfer divisor from HL pair to DE pair
LHLD 9052h    ; Load dividend from memory location C052
to HL pair.
LXI B,0000H ; Set BC to 0
loop: MOV A,L      ; A <- L [copy the lower 8 bits ]
SUB E       ; A = A - E [subtract the lower 8 bits ]
MOV L,A     ; L <- A
MOV A,H     ; A <- H [copy the higher 8 bits]
SBB D       ; Subtract the higher 8 bits with borrow.
MOV H,A     ; H <- A
INX B       ; Increment B
JNC loop      ; If not carry (which occurs if the
subtraction yielded a negative number) Jump to loop

DCX B       ; Since we over-counted B, decrement B
DAD D       ; Add DE to HL (makes it positive)
STORERESULT: SHLD 9054h   ; Puts remainder at C054
MOV A,C     ; A <- C
STA 9056h    ; Store lower 8 bits to memory location
MOV A,B     ; A <- B
STA 9057h   ; Store higher 8 bits to memory location
RST 05
```

Testing on the 8085 Simulator

The 8085 Simulator provides an extremely user friendly and graphical interface of the memory status at any time. It provides the programmer with various options:

1. One can write the 8085 Assembly program in a text editor provided by the simulator itself and not just upload files.
2. It provides the view of the memory in the form of a table.
3. It shows the status of the Assembled Program in a separate section.
4. One can execute the code step-wise or all at once.
5. It provides various options for storing data directly into the memory.

Testing on the MPS 85-3 Toolkit

The MPS-2 8085 Toolkit provides a physical board (hardware) with a small memory, an 8085 Microprocessor along with other components and a keyboard to operate the device/kit.

Steps that were followed to test the programs on the MPS-2 8085 Toolkit are given below.

1. Turned the Toolkit into 'Keyboard' Mode.
2. Entered the values of two operands into a set of memory locations using the Toolkit's keyboard.
3. Changed the Toolkit to 'Serial' Mode. In the serial mode, the Toolkit can connect connect with an external device using a 9-pin serial port. The Toolkit also provides its own Driver to interface with external devices.
4. The Driver was downloaded from the 'IITG Intranet Website' and installed on an external Desktop. The Computer was then connected with the Toolkit using through a 9-pin serial cable.
5. Using the command line tools provided by the Toolkit's driver, the assembly language programs were loaded into the Toolkit's memory.
6. The Toolkit was once again changed to the 'Keyboard' mode of operation.
7. Using the Toolkit's Keyboard interface, the program was executed and the results were tested for correctness.
8. Steps 6. and 7. were repeated for a number of test cases to ensure the correctness of the assembly language programs.

-------- End of Report -------