Qatar University

College of Engineering

Department of Computer Science and Engineering

CMPE263 Computer Architecture and Organization I
Course Project Report
Spring 2021

PROJECT TITLE SOFTWARE IMPLEMENTATION OF FLOATING-POINT ADD AND MULTIPLY OPERATIONS ON MANO'S COMPUTER

Submitted By:

1. Hani Jassin Jafer 201908619
2. Ahmed Abdelhamid 201907476
3. Taimoor Hussain 201904453

Table of Contents

Introduction 3
Floating-Point Calculations 3
  Floating point addition 3
  Unsigned integer multiplication subroutine 5
  Floating-point multiplication 6
Conclusion 9

Introduction

The project consists of two sections where each section represents a floating-point arithmetic calculation that is done by using an Assembly Language program. By implementing the Mano's instruction set architecture provided, two calculations are performed; Addition and Multiplication, with normalized floating-point results. For the multiplication part, instead of implementing subroutines to perform the calculation, we've used a different approach. The values used for input data are taken from the project document and have been tested with other input values

Floating-Point Calculations

Floating point addition

```
ORG 0

;Finding the difference between the input exponents

LDA BPTR I ;Load smaller exponent of B

CMA ;Complement the value

INC ;Increment

ADD APTR I ;Add with value of larger exponent of A

STA DIFF ;Store this result as the difference between the two exponents

CMA ;Complement the difference

INC ;Increment by 1

STA CTR ;Store this result as the negative of the difference


;Incrementing APTR and BPTR to point to the mantissa values

ISZ APTR

ISZ BPTR


LDA BPTR I ;Load the value of B mantissa

STA BADD ;Store in BADD to work with during the addition



LDA DIFF ;Loading diff of input exponents
```

```
SZA ;Skip IF block in case diff is 0

BUN IF ;Branch to IF when diff is not 0


;Adding A & B mantissas and storing the addition in C.

CNT, LDA BADD

ADD APTR I

ISZ CPTR

STA CPTR I


;Storing the exponent of A to C.

LDA A

STA C


;Normalizing the result incase theres a carry.

SZE

BUN NORM

CNTT, CLA



;After HLT

;Shifting B N bits to the right based on the exponent diff between A and B


IF, LDA BPTR I ;Load mantissa of input B

LOP, CLE ;Loop to shift right based on exponent difference of inputs

CIR ;Shift right

ISZ CTR ;Increment counter

BUN LOP ;Repeat loop if counter not 0

STA BADD ;Store result after shifts in BADD

BUN CNT ;Continue after all shifts are done (alignment)
```

```
;Normalizing C incase theres a carry by shifting the mantissa and incrementing the exponent

NORM, LDA CPTR I ;Load mantissa of addition result to AC
CIR ;Shift right
STA CPTR I ;Store modified value in position of C mantissa
LDA A ;Load exponent of input A
INC ;Increment by 1
STA C ;Store new value as exponent of result C
BUN CNTT ;Continue after normalization
```

Unsigned integer multiplication subroutine

Subroutine to perform unsigned integer multiplication from the textbook:

```
;Assembly Language Program: Multiplication
ORG 100
LOP, CLE ; Clear E
LDA Y ; Load multiplier
CIR ; Transfer multiplier bit to E
STA Y ; Store shifted multiplier
SZE ; Check if bit is zero
BUN ONE ; Bit is one; goto ONE
BUN ZRO ; Bit is zero; goto ZRO
ONE, LDA X ; Load multiplicand
ADD P ; Add to partial product
STA P ; Store partial product
CLE ; Clear E
ZRO, LDA X ; Load multiplicand
CIL ; Shift left
STA X ; Store shifted multiplicand
ISZ CTR ; Increment counter
BUN LOP ; Counter not zero; repeat loop
```

```
HLT ; Counter is zero; halt

CTR. DEC -8 ; This location serves as a counter X,

HEX 000F ; Multiplicand stored here

Y, HEX 000B ; Multiplier stored here

P, HEX 0 ; Product formed here

END
```

Floating-point multiplication

```
;Multiplying part.


;Load values of A and B mantissa to temp locations TMPA and TMPB

LDA APTR I

STA TMPA

LDA BPTR I

STA TMPB


;Shifting A and B mantissa to the right 8 times by looping

SHIFT, LDA SHFTCTR ;Load shift counter

LDA TMPA ;Load value of TMPA

CLE ;Clear E

CIR ;Shift right

STA TMPA ;Store value in TMPA

LDA TMPB ;Load value of TMPB

CLE ;Clear E

CIR ;Shift right

STA TMPB ;Store value in TMPB

ISZ SHFTCTR ;Increment shift counter

BUN SHIFT ;Repeat loop if shift counter is not 0
```

```
CLA ;Clear AC

ISZ DPTR ;Increment DPTR to point to mantissa of mutliplication result


;storing value of TMPB in BCTR to use for repeated addition

LDA TMPB

STA BCTR


;Loop to multiply A and B.

;Works by adding A to itself B times until B is 0.

;We Add A to itself instead of B.

;Since A > B and so, its much faster adding A to itself than the opposite.

LOOPM, LDA BCTR

SZA

BUN check


;Storing the exponent of A+B in D

LDA A

ADD B

STA D


;Normalizing the result of multiplication

NORMMULT, LDA DPTR I ;Load mantissa value of result D

AND MASK ;AND with MASK value to check if MSB is 1

SZA ;If MSB is 1, MASK will lead to 1, otherwise 0

BUN CNTMULT ;Continue as MSB is 1 and result is normalized

;In this stage, MASK lead to 0 and result needs to be normalized

LDA DPTR I ;Load mantissa of result

CLE ;Clear E

CIL ;Shift left

STA DPTR I ;Store mantissa after shift
```

```
LDA D ;Load exponent of result

ADD SUB ;Subtract 1 from exponent

STA D ;Store exponent after subtraction

BUN NORMMULT ;Continue loop


CNTMULT, HLT ;Program stops


;Multiplys A and B

;decrements B by 1 until B==0

;storing result in DPTR .


check, ADD SUB ;Subtract 1 from BCTR

STA BCTR ;Store value of BCTR after subtraction

LDA DPTR I ;Load current value of mantissa of result D

ADD TMPA ;Add mantissa of A to it

STA DPTR I ;Store result of addition as mantissa of D

BUN LOOPM ;Continue loop


;Input and other variables

ORG 2048


APTR, DEC 2052

BPTR, DEC 2054

CPTR, DEC 2056

DPTR, DEC 2058

A, DEC 5

HEX A000

B, DEC 2

HEX C000

C, DEC 0 ;Exponent of addition result stored here
```

```
DEC 0 ;Mantissa of addition result stores here

D, DEC 0 ;Exponent of multiplication result stored here

DEC 0 ;Mantissa of multiplication result stored here

DIFF, DEC 0 ;Difference between the 2 input exponents stored here

BADD, DEC 0 ;Temp loc to store B after shift for add

TMPA, DEC 0 ;Temp loc to store A after shift for mult

TMPB, DEC 0 ;Temp loc to store B after shift for mult

CTR, DEC 0 ;Negative value of the difference between input exponents, used as counter

MASK, HEX 8000 ;Value to mask using to check MSB=1

SHFTCTR, DEC -8 ;Shift counter to allow 8 shifts for multiplication

BCTR, DEC 0 ;B counter to check for 0

SUB, DEC -1  ;subtract

END
```

Conclusion

This project includes a program that calculates the floating-point addition and multiplication arithmetic on Mano's computer using the instruction sets given to us.

We have found out that calculation of floating-point arithmetic isn't as simple as compared to integer arithmetic as with integer we don't need to worry about decimal places. Therefore, there's no need to find out the difference of the exponent or shift the mantissa of the subtrahend but rather just add the operands together to get us the Result. The multiplication part however, isn't as easy to calculate as Mano's computer doesn't support multiplication as part of its instruction set which proved to be quite challenging, given that multiplication in the real world is easy to calculate. However, given the challenges we were able to calculate the multiplication of the operands by adding one of the operands to itself N times (or rather the second operand). Below are some screenshots and samples used to test this program.

Test 1:

A: Mantissa Hex A000 Exponent: 5

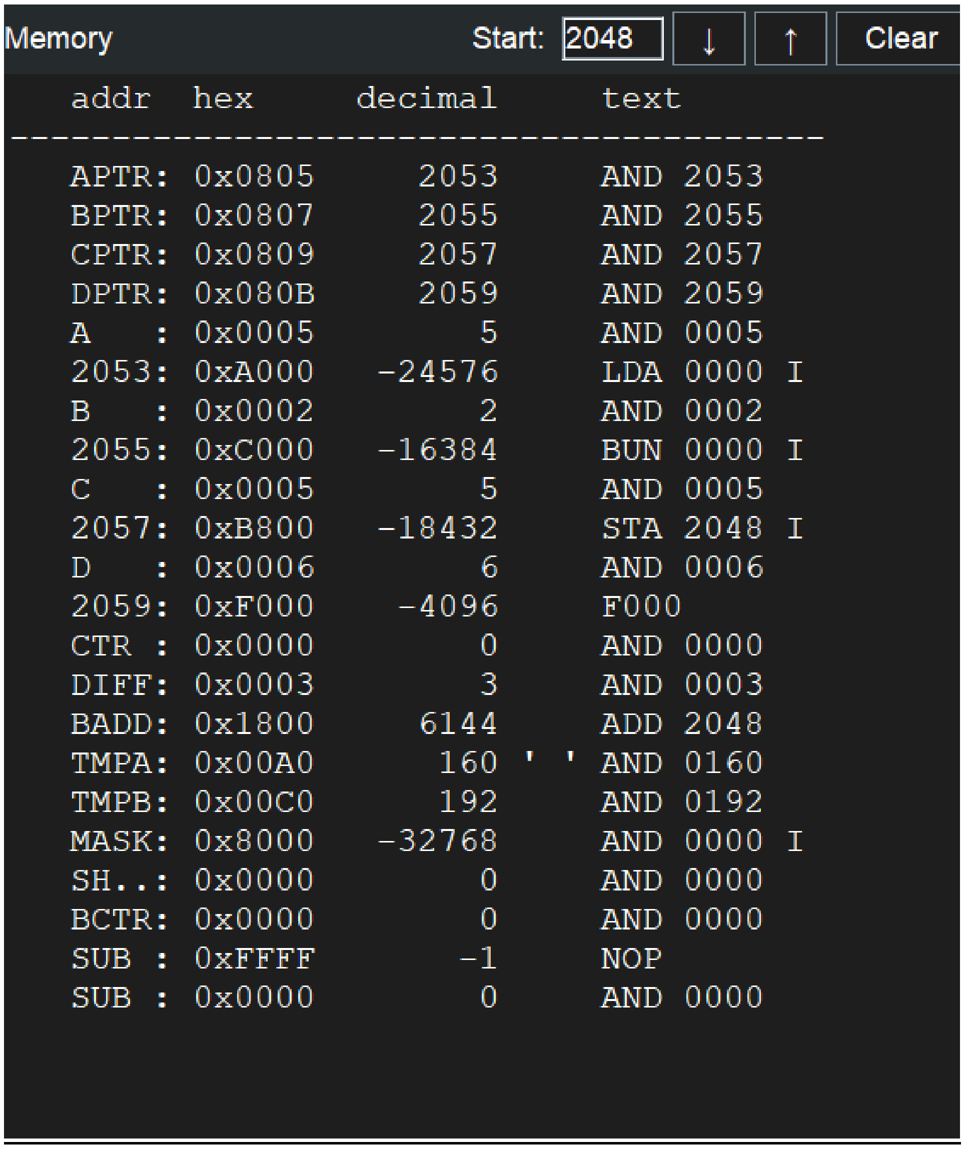
B: Mantissa Hex C000 Exponent:2

Expected output:

C: Mantissa Hex B800 Exponent: 5

D: Mantissa Hex F000 Exponent: 6

Output:



```
   addr   hex      decimal      text
----------------------------------------
   APTR: 0x0805     2053     AND 2053
   BPTR: 0x0807     2055     AND 2055
   CPTR: 0x0809     2057     AND 2057
   DPTR: 0x080B     2059     AND 2059
   A   : 0x0005        5     AND 0005
   2053: 0xA000   -24576     LDA 0000 I
   B   : 0x0002        2     AND 0002
   2055: 0xC000   -16384     BUN 0000 I
   C   : 0x0005        5     AND 0005
   2057: 0xB800   -18432     STA 2048 I
   D   : 0x0006        6     AND 0006
   2059: 0xF000    -4096     F000
   CTR : 0x0000        0     AND 0000
   DIFF: 0x0003        3     AND 0003
   BADD: 0x1800     6144     ADD 2048
   TMPA: 0x00A0      160 ' ' AND 0160
   TMPB: 0x00C0      192     AND 0192
   MASK: 0x8000   -32768     AND 0000 I
   SH..: 0x0000        0     AND 0000
   BCTR: 0x0000        0     AND 0000
   SUB : 0xFFFF       -1     NOP
   SUB : 0x0000        0     AND 0000
```

Test 2:

A: Mantissa Hex A000 Exponent: 5

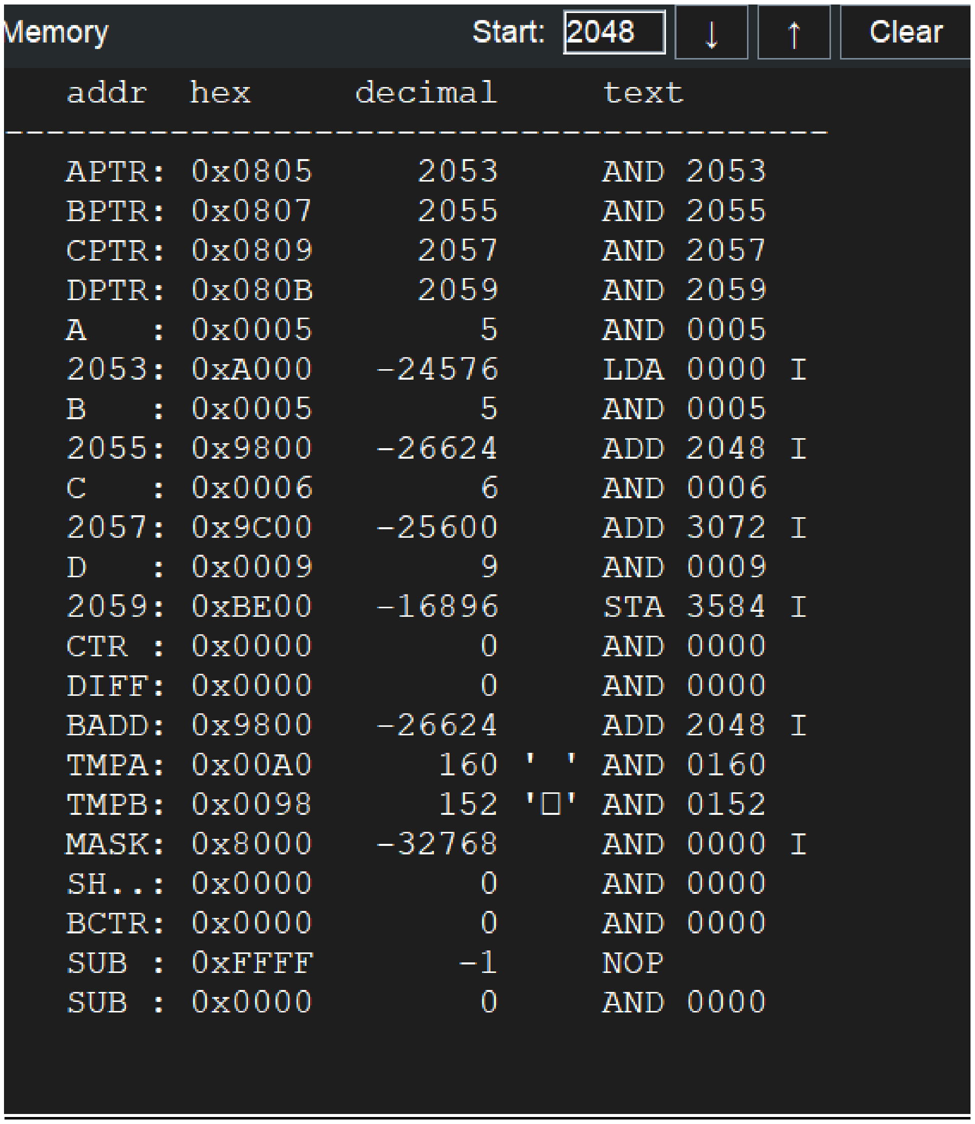
B: Mantissa Hex 9800 Exponent: 5

Expected output:

C: Mantissa Hex 9C00 Exponent: 6

D: Mantissa Hex BE00 Exponent: 9

Output:



```
   addr  hex      decimal      text
----------------------------------------
   APTR: 0x0805      2053      AND 2053
   BPTR: 0x0807      2055      AND 2055
   CPTR: 0x0809      2057      AND 2057
   DPTR: 0x080B      2059      AND 2059
   A   : 0x0005         5      AND 0005
   2053: 0xA000    -24576      LDA 0000 I
   B   : 0x0005         5      AND 0005
   2055: 0x9800    -26624      ADD 2048 I
   C   : 0x0006         6      AND 0006
   2057: 0x9C00    -25600      ADD 3072 I
   D   : 0x0009         9      AND 0009
   2059: 0xBE00    -16896      STA 3584 I
   CTR : 0x0000         0      AND 0000
   DIFF: 0x0000         0      AND 0000
   BADD: 0x9800    -26624      ADD 2048 I
   TMPA: 0x00A0       160 ' '  AND 0160
   TMPB: 0x0098       152 '□'  AND 0152
   MASK: 0x8000    -32768      AND 0000 I
   SH..: 0x0000         0      AND 0000
   BCTR: 0x0000         0      AND 0000
   SUB : 0xFFFF        -1      NOP
   SUB : 0x0000         0      AND 0000
```