Contents

1 README

```
roigreenberg,inbaravni
===============================================================================
Roi Greenberg, ID 30557123, roi.greenberg@mail.huji.ac.il
Inbar Avni, ID 201131760, inbar.avni@mail.huji.ac.il
===============================================================================

                           Project 4 - Machine Language Programming
                           ----------------------------------------


Submitted Files
---------------
README - This file.
Mult.asm - this program computes the value R0*R1 and stores the result in R2.
Fill.asm - the program runs an infinite loop that listens to the keyboard input.
       When any key is pressed, the program blackens the screen. When no
       key is pressed, the program clears the screen.
Sort.asm - The program should sort the array starting at the address in R14
       with length specified in R15.
Divide.asm - The program input will be at R13,R14 while the result R13/R14
         will be store at R15.


Remarks
-------
:)
```

2 divide/Divide.asm

```
//set bit at MSB of the denominator
@16384
D=A
@currBit
M=D
(Setbit)
    @R13
    D=M
    @currBit
    D=D&M
    M=M>>
    @Setbit
    D;JEQ

//fix redundent shift
@currBit
M=M<<

// init result
@R15
M=0

@tmp
M=0

(Loop)
    @tmp
    M=M<<
    //check the current bit at the denominator
    @R13
    D=M
    @currBit
    D=D&M
    //skip if bit=0
    @Continue
    D;JEQ
    //add temp the next bit
    D=1
    @tmp
    M=D|M

(Continue)
    //check if tmp - R14 > 0
    @tmp
    D=M
    @R14
    D=D-M
    //skip if false
    @Endif
    D;JLT
    @tmp
    M=D
    @currBit
    D=M
    //save current result
    @R15
    M=D|M
(Endif)
    //end if curr bit is 0
    @currBit
    D=M
    @End
    D;JEQ

    @currBit
    M=M>>
    @Loop
    0;JMP

(End)

```

3 fill/Fill.asm

```
// This file is part of www.nand2tetris.org
// and the book "The Elements of Computing Systems"
// by Nisan and Schocken, MIT Press.
// File name: projects/04/Fill.asm

// Runs an infinite loop that listens to the keyboard input.
// When a key is pressed (any key), the program blackens the screen,
// i.e. writes "black" in every pixel. When no key is pressed, the
// program clears the screen, i.e. writes "white" in every pixel.



(Loopkbd)
    //save the value of if keyboard is in use
    @KBD
    D=M

    // keyboard is in use - blacken screen
    @Loopscrb
    D@JGT

    // keyboard isn't in use - whiten screen
    @Loopscrw
    0;JMP

(Loopscrb)
    // init variable for screen location
    @SCREEN
    D=A
    @i
    M=D

(Loop2)
    // running for all screen cells and blacken them
    @i
    A=M
    M=-1

    @i
    M=M+1
    D=M

    @KBD
    D=D-A

    @Loop2
    D;JLT

    @Loopkbd
    0;JMP

(Loopscrw)
    // init variable for screen location
    @SCREEN
    D=A
    @i
    M=D

(Loop3)
```

```
// running for all screen cells and blacken them
@i
A=M
M=0

@i
M=M+1
D=M
@KBD
D=D-A
@Loop3
D;JLT
@Loopkbd
0;JMP
```

4 mult/Mult.asm

```
// This file is part of www.nand2tetris.org
// and the book "The Elements of Computing Systems"
// by Nisan and Schocken, MIT Press.
// File name: projects/04/Mult.asm

// Multiplies R0 and R1 and stores the result in R2.
// (R0, R1, R2 refer to RAM[0], RAM[1], and RAM[2], respectively.)

    //save R0 in tmp
    @R0
    D=M
    @tmp
    M=D

    //set R2 to 0
    @R2
    M=0

    //initialize index i
    @i
    M=1

(Loop)
    //take the i's bit of R1
    @R1
    D=M
    @i
    D=D&M

    //continue if D is not zero
    @Endif
    D;JEQ

    //add R2 the value in tmp
    @tmp
    D=M
    @R2
    M=D+M


(Endif)

    //shift left tmp and i
    @tmp
    D=M
    M=D+M

    @i
    D=M
    M=D+M
    D=M

    //while i<=32768
    @32768
    D=A-D
    @Loop
    D;JGE

```

5 sort/Sort.asm

```
//init the value to length
@R15
D=M
@length
M=D


(Loop)
    // save the starting location of the array
    @R14
    D=M
    @currentLocation
    M=D

    // updating the remained length to sort
    @length
    M=M-1
    D=D+M
    @endLocation
    M=D


    // init the value for the first node in array
    @currentLocation
    A=M
    D=M
    @currentVal
    M=D





// running for all the array values
(Mainloop)
    // updating the location in array, and saving the value of 'next'
    @currentLocation
    M=M+1
    A=M
    D=M
    @nextVal
    M=D

    // check if need to swap
    @currentVal
    D=M-D
    @Swap
    D;JLT

(Continue)
    // updating the currentVal
    @currentLocation
    A=M
    D=M
    @currentVal
    M=D

    @currentLocation
    D=M
    @endLocation
    D=M-D
    // back to running over the array till the end of it
    @Mainloop
    D;JGT
    //go to outer loop
    @length
    D=M-1
    @Loop
    D;JGT
    @End
    0;JMP

(Swap)
    @currentVal
    D=M
    @currentLocation
    A=M
    M=D

    @nextVal
    D=M
    @currentLocation
    A=M-1
    M=D

    @Continue
    0;JMP

(End)
```