Contents

1	README	2
2	divide/Divide.asm	3
3	fill/Fill.asm	5
4	mult/Mult.asm	7
5	sort/Sort.asm	8

1 README

```
roigreenberg,inbaravni
1
   Roi Greenberg, ID 30557123, roi.greenberg@mail.huji.ac.il
    Inbar Avni, ID 201131760, inbar.avni@mail.huji.ac.il
4
    _____
                             Project 4 - Machine Language Programming
8
9
10
   Submitted Files
11
12
   README - This file.
    Mult.asm - this program computes the value R0*R1 and stores the result in R2.
14
   Fill.asm - the program runs an infinite loop that listens to the keyboard input.
15
          When any key is pressed, the program blackens the screen. When no
16
          key is pressed, the program clears the screen.
17
    Sort.asm - The program should sort the array starting at the address in R14
18
          with length specified in R15.
19
   Divide.asm - The program input will be at R13,R14 while the result R13/R14
20
21
            will be store at R15.
22
23
24
   Remarks
25
   :)
```

2 divide/Divide.asm

```
//set bit at MSB of the denominator
3
    D=A
    @currBit
    M=D
    (Setbit)
        D=M
9
        @currBit
        D=D&M
10
11
        M=M>>
        @Setbit
12
        D;JEQ
13
15
    //fix redundent shift
    @currBit
16
    M=M<<
18
    // init result
19
20
    M=O
21
22
23
24
    M=O
26
    (Loop)
^{27}
         @tmp
28
        //check the current bit at the denominator
29
30
        @R13
31
        D=M
        @currBit
32
        D=D&M
        //skip if bit=0
34
        @Continue
35
        D;JEQ
36
        //add temp the next bit
37
38
        D=1
        @tmp
39
        M=D \mid M
40
41
     (Continue)
42
        // {\rm check} if tmp - R14 > 0
43
44
        D=M
45
        @R14
46
47
        D=D-M
        //skip if false
48
         @Endif
        D; JLT
50
51
        @tmp
        M=D
52
        @currBit
53
54
        D=M
        //save current result
        @R15
56
        M=D \mid M
58
        //end if curr bit is 0
```

```
@currBit
60
61
         D=M
         @End
62
63
         D;JEQ
64
         @currBit
65
         M=M>>
@Loop
O;JMP
66
67
68
69
     (End)
70
71
```

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.
10
11
12
     (Loopkbd)
13
14
         //save the value of if keyboard is in use
15
        D=M
16
17
         // keyboard is in use - blacken screen
18
19
         @Loopscrb
20
21
         // keyboard isn't in use - whiten screen
22
         @Loopscrw
23
         O;JMP
24
25
     (Loopscrb)
26
         // init variable for screen location
27
28
         @SCREEN
        D=A
29
30
         @i
        M=D
31
32
         // running for all screen cells and blacken them
34
         @i
35
         A=M
36
        M=-1
37
38
39
        M=M+1
40
41
        D=M
42
         @KBD
43
        D=D-A
44
45
46
         @Loop2
47
        D;JLT
48
         @Loopkbd
         O;JMP
50
51
     (Loopscrw)
52
         // init variable for screen location
53
         @SCREEN
54
55
56
         @i
57
        M=D
58
    (Loop3)
```

```
\ensuremath{//} running for all screen cells and blacken them
60
61
          @i
          A=M
62
63
          M=O
64
          @i
65
          M=M+1
66
67
          D=M
          @KBD
68
          D=D-A
69
          @Loop3
D;JLT
@Loopkbd
O;JMP
70
71
72
73
```

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 \ensuremath{\text{RO}} and \ensuremath{\text{R1}} and stores the result in \ensuremath{\text{R2}}.
     // (RO, R1, R2 refer to RAM[0], RAM[1], and RAM[2], respectively.)
 9
          //save RO in tmp
10
          @RO
          D=M
11
12
          @tmp
          M=D
13
14
15
          //set R2 to 0
          @R2
16
17
          M=0
18
          //initialize index i
19
20
          M=1
21
22
23
          \frac{1}{1}/take the i's bit of R1
24
25
          @R1
          D=M
26
          0i
27
          D=D&M
29
          //continue if D is not zero
30
31
          @Endif
          D;JEQ
32
33
34
          //add R2 the value in tmp
          @tmp
35
          D=M
          @R2
37
          M=D+M
38
40
     (Endif)
41
42
          //shift left tmp and i
43
44
          D=M
45
          M=D+M
46
          @i
48
49
          D=M
          M=D+M
50
          D=M
51
          //while i<=32768
53
          @32768
54
          D=A-D
          @Loop
56
          D;JGE
57
```

5 sort/Sort.asm

```
//init the value to length
2
    D=M
3
    @length
4
5
    M=D
    (Loop)
9
         // save the starting location of the array
        @R14
10
11
        D=M
         @currentLocation
12
        M=D
13
14
15
        // updating the remained length to sort
        @length
16
17
        M=M-1
        D=D+M
18
         {\tt @endLocation}
19
20
21
22
        // init the value for the first node in array
23
         @currentLocation
24
25
        D=M
26
         @currentVal
^{27}
28
29
30
31
32
    // running for all the array values
34
     (Mainloop)
35
         // updating the location in array, and saving the value of 'next'
36
         @currentLocation
37
        M=M+1
38
        A=M
39
        D=M
40
41
         @nextVal
42
43
44
        // check if need to swap
        @currentVal
45
        D=M-D
46
47
         @Swap
        D; JLT
48
     (Continue)
50
         // updating the currentVal
51
         @currentLocation
52
         A=M
53
54
        D=M
55
         @currentVal
56
        @currentLocation
58
        D=M
59
```

```
{\tt @endLocation}
60
61
         D=M-D
         \ensuremath{//} back to running over the array till the end of it
62
         {\tt @Mainloop}
63
64
         D; JGT
         //go to outer loop
65
         @length
66
67
         D=M-1
68
         @Loop
69
         D;JGT
         @End
70
         0;JMP
71
72
     (Swap)
73
74
         @currentVal
         D=M
75
76
         {\tt @currentLocation}
77
         A=M
         M=D
78
79
         @nextVal
80
         D=M
81
82
         {\tt @currentLocation}
         A=M-1
83
         M=D
84
85
         @Continue
86
87
         0;JMP
88
     (End)
89
```