

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

```
1 roigreenberg,inbaravni
2 =====
3 Roi Greenberg, ID 30557123, roi.greenberg@mail.huji.ac.il
4 Inbar Avni, ID 201131760, inbar.avni@mail.huji.ac.il
5 =====
6
7                 Project 4 - Machine Language Programming
8                 -----
9
10
11 Submitted Files
12 -----
13 README - This file.
14 Mult.asm - this program computes the value R0*R1 and stores the result in R2.
15 Fill.asm - the program runs an infinite loop that listens to the keyboard input.
16             When any key is pressed, the program blackens the screen. When no
17             key is pressed, the program clears the screen.
18 Sort.asm - The program should sort the array starting at the address in R14
19             with length specified in R15.
20 Divide.asm - The program input will be at R13,R14 while the result R13/R14
21             will be store at R15.
22
23
24 Remarks
25 -----
26 :)
```

2 divide/Divide.asm

```
1  //set bit at MSB of the denominator
2  @16384
3  D=A
4  @currBit
5  M=D
6  (Setbit)
7      @R13
8      D=M
9      @currBit
10     D=D&M
11     M=M>>
12     @Setbit
13     D;JEQ
14
15 //fix redundant shift
16 @currBit
17 M=M<<
18
19 // init result
20 @R15
21 M=0
22
23 @tmp
24 M=0
25
26 (Loop)
27     @tmp
28     M=M<<
29     //check the current bit at the denominator
30     @R13
31     D=M
32     @currBit
33     D=D&M
34     //skip if bit=0
35     @Continue
36     D;JEQ
37     //add temp the next bit
38     D=1
39     @tmp
40     M=D|M
41
42 (Continue)
43     //check if tmp - R14 > 0
44     @tmp
45     D=M
46     @R14
47     D=D-M
48     //skip if false
49     @Endif
50     D;JLT
51     @tmp
52     M=D
53     @currBit
54     D=M
55     //save current result
56     @R15
57     M=D|M
58 (Endif)
59     //end if curr bit is 0
```

```
60      @currBit
61      D=M
62      @End
63      D;JEQ
64
65      @currBit
66      M=M>>
67      @Loop
68      0;JMP
69
70 (End)
71
```

3 fill/Fill.asm

```
1  // This file is part of www.nand2tetris.org
2  // and the book "The Elements of Computing Systems"
3  // by Nisan and Schocken, MIT Press.
4  // File name: projects/04/Fill.asm
5
6  // Runs an infinite loop that listens to the keyboard input.
7  // When a key is pressed (any key), the program blackens the screen,
8  // i.e. writes "black" in every pixel. When no key is pressed, the
9  // program clears the screen, i.e. writes "white" in every pixel.
10
11
12
13 (Loopkbd)
14     //save the value of if keyboard is in use
15     @KBD
16     D=M
17
18     // keyboard is in use - blacken screen
19     @Loopscrb
20     D@JGT
21
22     // keyboard isn't in use - whiten screen
23     @Loopscrw
24     0;JMP
25
26 (Loopscrb)
27     // init variable for screen location
28     @SCREEN
29     D=A
30     @i
31     M=D
32
33 (Loop2)
34     // running for all screen cells and blacken them
35     @i
36     A=M
37     M=-1
38
39     @i
40     M=M+1
41     D=M
42
43     @KBD
44     D=D-A
45
46     @Loop2
47     D;JLT
48
49     @Loopkbd
50     0;JMP
51
52 (Loopscrw)
53     // init variable for screen location
54     @SCREEN
55     D=A
56     @i
57     M=D
58
59 (Loop3)
```

```
60      // running for all screen cells and blacken them
61      @i
62      A=M
63      M=0
64
65      @i
66      M=M+1
67      D=M
68      @KBD
69      D=D-A
70      @Loop3
71      D;JLT
72      @Loopkbd
73      0;JMP
```

4 mult/Mult.asm

```
1 // This file is part of www.nand2tetris.org
2 // and the book "The Elements of Computing Systems"
3 // by Nisan and Schocken, MIT Press.
4 // File name: projects/04/Mult.asm
5
6 // Multiplies R0 and R1 and stores the result in R2.
7 // (R0, R1, R2 refer to RAM[0], RAM[1], and RAM[2], respectively.)
8
9 //save R0 in tmp
10 @R0
11 D=M
12 @tmp
13 M=D
14
15 //set R2 to 0
16 @R2
17 M=0
18
19 //initialize index i
20 @i
21 M=1
22
23 (Loop)
24 //take the i's bit of R1
25 @R1
26 D=M
27 @i
28 D=D&M
29
30 //continue if D is not zero
31 @Endif
32 D;JEQ
33
34 //add R2 the value in tmp
35 @tmp
36 D=M
37 @R2
38 M=D+M
39
40
41 (Endif)
42
43 //shift left tmp and i
44 @tmp
45 D=M
46 M=D+M
47
48 @i
49 D=M
50 M=D+M
51 D=M
52
53 //while i<=32768
54 @32768
55 D=A-D
56 @Loop
57 D;JGE
58
```

5 sort/Sort.asm

```
1  //init the value to length
2  @R15
3  D=M
4  @length
5  M=D
6
7
8  (Loop)
9      // save the starting location of the array
10     @R14
11     D=M
12     @currentLocation
13     M=D
14
15     // updating the remained length to sort
16     @length
17     M=M-1
18     D=D+M
19     @endLocation
20     M=D
21
22
23     // init the value for the first node in array
24     @currentLocation
25     A=M
26     D=M
27     @currentVal
28     M=D
29
30
31
32
33
34 // running for all the array values
35 (Mainloop)
36     // updating the location in array, and saving the value of 'next'
37     @currentLocation
38     M=M+1
39     A=M
40     D=M
41     @nextVal
42     M=D
43
44     // check if need to swap
45     @currentVal
46     D=M-D
47     @Swap
48     D;JLT
49
50 (Continue)
51     // updating the currentVal
52     @currentLocation
53     A=M
54     D=M
55     @currentVal
56     M=D
57
58     @currentLocation
59     D=M
```



```

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

```