**Foundations of Computer Science – Exercise 5**

2.

a) LOAD 100: ACC <- (100): ACC=105

STORE 108: (108) <- ACC: (108) = 115

ADD 101: ACC <- ACC + (101): ACC = 211

STORE 109: (109) <- ACC: (109) = 211

* ACC = 211, (108) = 115, (109) = 211

b) LOAD 109: ACC <- (109): ACC = 114

STORE 100: (100) <- ACC: (100) = 114

LOAD 108: ACC <- (108): ACC = 113

STORE 101: (101) <- ACC: (101) = 113

LOAD 107: ACC <- (107): ACC = 112

ADD 103: ACC <- ACC + (103): ACC = 220

STORE 104: (104) <- ACC: (104) = 220

* ACC = 220, (100) = 114, (101) = 113, (104) = 220

3.

|  |  |  |
| --- | --- | --- |
| **Address** | **Command** | **Explanation** |
| 101 | LOAD 114 | ACC <- (114): ACC = 5 |
| 102 | JUMPZERO 117 | If ACC = 0, jump to (117) [STOP] |
| 103 | STORE 115 | (115) <- ACC: (115) = 5 |
| 104 | SUBTRACT 116 | ACC <- ACC – (116): ACC = 4 |
| 105 | STORE 114 | (114) <- ACC: (114) = 4 |
| 106 | LOAD 114 | ACC <- (114): ACC = 4 |
| 107 | JUMPZERO 117 | If ACC = 0, jump to (117) [STOP] |
| 108 | MULTIPLY 115 | ACC ← ACC \* (115): ACC = 20 |
| 109 | STORE 115 | (115) <- ACC: (115) = 20 |
| 110 | LOAD 114 | ACC <- (114): ACC = 4 |
| 111 | SUBTRACT 116 | ACC <- ACC – (116): ACC = 3 |
| 112 | STORE 114 | (114) <- ACC: (114) = 3 |
| 113 | JUMP 106 | Jump to (106) |
| 114 | 5 | Function value |
| 115 | 1 | Function value |
| 116 | 1 | Number constant |

114: 5 – 4 – 3 – 2 – 1 – 0

115: 1 – 5 – 20 – 60 – 120 – 120

The program calculates the value of the sequence:

1 x 5 = 5

5 x 4 = 20

20 x 3 = 60

60 x 2 = 120

120 x 1 = 120

4.

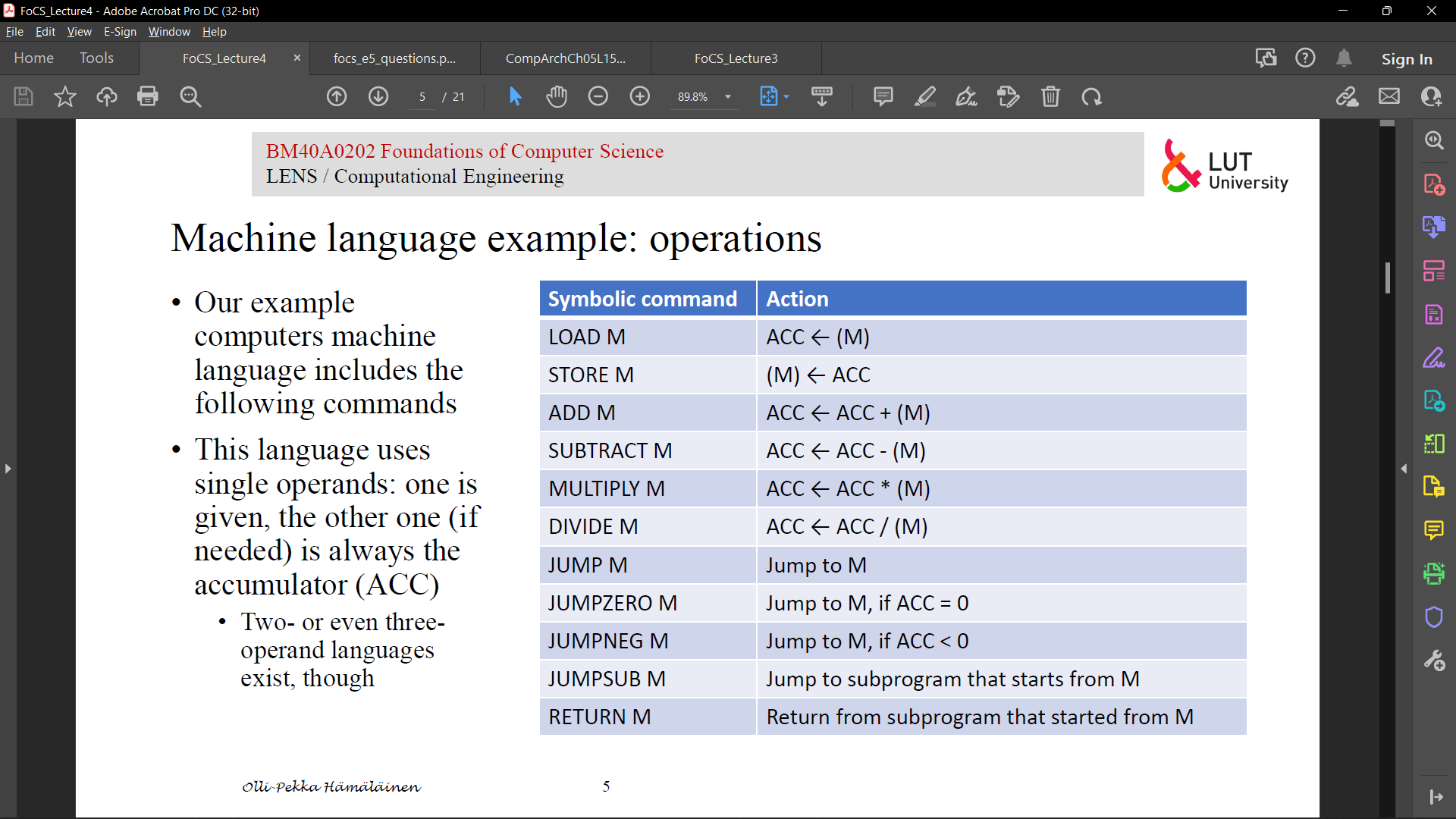
a)

|  |  |  |
| --- | --- | --- |
| **Address** | **Command** | **Explanation** |
| 371 | LOAD 383 | Load 1 to ACC |
| 372 | STORE 382 | Store 1 as initial value of x |
| 373 | LOAD 381 | Load n to ACC |
| 374 | JUMPZERO 384 | If n = 0, jump to (115) |
| 375 | MULTIPLY 382 | Multiply n by x |
| 376 | STORE 381 | Save new value of n |
| 377 | LOAD 382 | Load value x to ACC |
| 378 | SUBTRACT 383 | ACC = x - 1 |
| 379 | STORE 382 | Save new value of x |
| 380 | JUMP 373 | Jump to beginning of iteration |
| 381 | n | Parameter |
| 382 | 1 | Function value x |
| 383 | 1 | Number constant |

b)

|  |  |  |
| --- | --- | --- |
| **Address** | **Command** | **Explanation** |
| 371 | LOAD 386 | Load 1 to ACC |
| 372 | STORE 385 | Store 1 as initial value of x |
| 373 | JUMPZERO 387 | If n = 0, jump to (386) |
| 374 | JUMPSUB 379 | Execute subprogram |
| 375 | LOAD 385 | Load value x to ACC |
| 376 | SUBTRACT 386 | ACC = x - 1 |
| 377 | STORE 385 | Save new value of x |
| 378 | JUMP 373 | Jump to beginning of iteration |

|  |  |  |
| --- | --- | --- |
| **Address** | **Command** | **Explanation** |
| 379 | 0 / 375 | Begin / Return |
| 380 | LOAD 384 | Load n to ACC |
| 381 | MULTIPLY 385 | Multiply n by x |
| 382 | STORE 384 | Save new value of n |
| 383 | RETURN 379 | Return to main program |
| 384 | n | Parameter |
| 385 | 1 | Value x |
| 386 | 1 | Number constant |

5. *I don’t really understand what I am supposed to do in this task, but here is my take:*

From the course “Fundamental of Computer Science”, University of Birmingham:

