CS3308 Exercise 6a

Kalen Williams

27 October 2016

(1)

Memory — Instruction — source — destination
00 60 08 00 Load memory08 with value of 10 into register00
01 50 00 09 Read in value from keyboard. Save value to memory09
02 71 09 00 Subtracts memory09 from register 00. Result is stored in register00
03 81 00 05 Skip to memory05 if register00 is negative
04 80 00 01 Set program counter to memory01
05 51 00 00 Displays value of register00
06 83 00 00 Stops the program
07 00 00 00 Unused in this program
08 00 00 10 Holds decimal value of 10
09 00 00 00 Holds user input

I don't see how this program deals with arrays. To me it seems that this program reads in user input and subtracts 10 from it. If $userInput - 10 < 0$ the program outputs the value of $userInput - 10$ and halts. If $userInput - 10 \geq 0$ the program loops until a value ≤ 9 is entered by user. After the disuccsion in class about identifying a bug I am going to say this program relates to arrays in that the memory you store the userInput in is supposed to be incremented every new loop so we have several consecutive memory locations storing input numbers.

---

(2)

```
00 60 14 03 // puts constant 10 in register03 01 50 00 15 //read userinput save to memory98
02 60 98 00 //load memory98 to register00
03 81 00 90 //Jump to halt if negative is entered
04 71 03 00 //subtract 10 from counter to see if we should exit
05 82 00 90 //Skip to output and halt if 10 numbers have been entered
06 70 00 04 //add register 00(userInput) to 04(sum) and place result in register 04
07 60 13 02 //load constant one into registter 02
08 70 02 01 //add one to counter
09 61 04 16 //store sum register to sum memory
10 80 00 01 // start over at memory01
11 51 16 00 //write memory99, the sum
12 83 00 00 // Halt
13 00 00 01 //Constant 1
14 00 00 10 //Constant 10
15 00 00 00 //this is the counter
16 00 00 00 //This is the sum
```

---

③

```
00 50 00 04 //Read in from keyboard store in USERINPUT
01 70 04 00 //Add USERINPUT to SUM. Store result in SUM
02 70 05 01 //Add CONSTANT 1 to SUM COUNTER
03 71 01 03 //Subtract CONSTANT 7 from SUM COUNTER
04 82 03 07 //If CONSTANT 7 - 7 = 0 GO TO XX(DIVIDE LOOP)
05 60 08 03 //Set Register 3 back to 7
06 80 00 00 //Continue getting userInput
07 60 08 03 //Set Register 3 back to 7
08 71 03 00 //Subtract 7 from SUM
09 70 05 02 //Add one to divide counter
10 81 00 14 //If sum is negative go to display
11 82 00 14 //If sum is 0 go to display
12 80 00 07 //Loop back to subtract 7 from SUM
13 61 02 16 //Store DIVIDECOUNT register into DivideCount memory
14 51 16 00 //Write DivideCount
15 83 00 00 // halt program
16 00 00 00 // Contains memory for divide counter
17 00 00 07 // Contains contant 7 memory
```

Register 00 = SUM
Register 01 = SUM COUNTER
Register 02 = DIVIDE COUNTER
Register 03 = CONSTANT 7
Register 04 = USERINPUT
Register 05 = CONSTANT 1
Register 06 = SUM RESULT
Memory 08 = Constant 7
Memory 09 = DivideCounter

---

④

```
00 60 99 02 // Register02 = constant 1
01 50 00 16 // Store count of numbers to get in NUMCOUNT
02 60 16 00 //Load NUMCOUNT into register00
03 50 00 03 //Read userInput into Register03
04 60 03 01 //Set first entry as current min
05 71 02 00 //Subtract 1 from numcount
06 82 00 13 //If got enough input display and halt
07 50 00 03 //get userInput
08 71 02 00 //Subtract 1 from numcount
09 71 01 03 //Subtract the current min from userInput
10 81 03 06 //Get more userInput if didn't enter new max
11 61 03 01 //Store userInput as current min
12 80 00 06 //get more user input
13 51 02 00 // display Max number
14 83 00 00 // Halt
15 00 00 01 // Constant One
16 00 00 00 //memory for total number of numbers to get
```

Register 00 = NUMCOUNT
Register 01 = CURRENTMax
Register 02 = CONSTANT ONE
Regiser 03 = UserInput