COMPUTER SCIENCE 110

**INTRODUCTION TO COMPUTER SCIENCE**

**HOMEWORK 4 (Python Multimedia)**

# Spring 2018

**Assigned**: Mon. March 12 **Due**: Fri. April 6 by 4pm

**Note**: You must answer Questions 3 and 4 on the printed assignment sheets and turn them in. All other questions must be answered on separate sheets of paper.

BD14710_

**Part I. (50 points) Von Neumann Architecture and Assembly/Machine Languages**

1. (4 pts.) Consider the following structure of the instruction register.

|  |  |
| --- | --- |
| opcode | address1 |

8 bits 10 bits

* 1. What is the maximum number of operation codes that can be recognized and executed by the processor? i.e., how many different opcodes can there be, maximum?
  2. What is the maximum memory size, in bytes, on this machine? Assume that each addressable memory cell is 3 bytes in size.

1. a. (4 pts.) Convert the following code into assembly language code. Start your code at memory cell 0 and assume that variables w, x, y, and z are stored in memory cells 101, 102, 103, and 104 respectively. If you like, you can use b as a temporary variable stored in memory cell 106.

**y = y + 1**

**w = x – (y + z)**

b. (8 pts.) Convert the following code into assembly language code. Assume that variables w and x are stored in memory cells 101 and 102 respectively:

**w = input()**

**x = input()**

**if w > x:**

**w = 0**

**w = w - 2**

**print w**

**else:**

**x = x \* 2**

**print x**

1. (8 Points) Answer the questions in this section starting with the values below. NOTE: Assume all of these **problems are sequential. Each problem (a through d) builds on the memory and register values of the one before.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Registers** | |  | **Memory Address** | **Memory Contents** |
| **PC** | 5 |  | #100 | 0 |
| **R** | 2 |  | #101 | 17 |
| **IR** | CLEAR #100 |  | #102 | 10 |
| **CCR** | 000 |  | #103 | 9 |

1. Fill in the values of the registers and memory cells after the following command is executed:

**INCREMENT #103**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Registers** | |  | **Memory Address** | **Memory Contents** |
| **PC** | **6** |  | #100 |  |
| **R** |  |  | #101 |  |
| **IR** |  |  | #102 |  |
| **CCR** |  |  | #103 |  |

1. Fill in the values of the registers and memory cells after the following command is executed:

**SUBTRACT #103**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Registers** | |  | **Memory Address** | **Memory Contents** |
| **PC** | **7** |  | #100 |  |
| **R** |  |  | #101 |  |
| **IR** |  |  | #102 |  |
| **CCR** |  |  | #103 |  |

1. Fill in the values of the registers and memory cells after the following command is executed:

**STORE #101**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Registers** | |  | **Memory Address** | **Memory Contents** |
| **PC** | **8** |  | #100 |  |
| **R** |  |  | #101 |  |
| **IR** |  |  | #102 |  |
| **CCR** |  |  | #103 |  |

1. Fill in the values of the registers and memory cells after the following command is executed:

**COMPARE #101**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Registers** | |  | **Memory Address** | **Memory Contents** |
| **PC** | **9** |  | #100 |  |
| **R** |  |  | #101 |  |
| **IR** |  |  | #102 |  |
| **CCR** |  |  | #103 |  |

1. (8 Points) Answer the questions in this section starting with the values below. **NOTE: Assume all of these problems are sequential. Each problem (a through d) builds on the memory and register values of the one before.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Registers** | |  | **Memory Address** | **Memory Contents** |
| **PC** | 17 |  | #100 | 111 |
| **R** | 1492 |  | #101 | 1812 |
| **IR** | CLEAR #103 |  | #102 | 1776 |
| **CCR** | 100 |  | #103 | 0 |

1. Fill in the values of the registers and memory cells after the following command is executed:

**LOAD #101**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Registers** | |  | **Memory Address** | **Memory Contents** |
| **PC** |  |  | #100 |  |
| **R** |  |  | #101 |  |
| **IR** |  |  | #102 |  |
| **CCR** |  |  | #103 |  |

1. Fill in the values of the registers and memory cells after the following command is executed:

**SUBTRACT #100**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Registers** | |  | **Memory Address** | **Memory Contents** |
| **PC** |  |  | #100 |  |
| **R** |  |  | #101 |  |
| **IR** |  |  | #102 |  |
| **CCR** |  |  | #103 |  |

1. Fill in the values of the registers and memory cells after the following command is executed:

**COMPARE #102**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Registers** | |  | **Memory Address** | **Memory Contents** |
| **PC** |  |  | #100 |  |
| **R** |  |  | #101 |  |
| **IR** |  |  | #102 |  |
| **CCR** |  |  | #103 |  |

1. Fill in the values of the registers and memory cells after the following command is executed:

**JUMPNEQ #42**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Registers** | |  | **Memory Address** | **Memory Contents** |
| **PC** |  |  | #100 |  |
| **R** |  |  | #101 |  |
| **IR** |  |  | #102 |  |
| **CCR** |  |  | #103 |  |

1. (4 pts.) Convert the following assembly language code into Python code; assume that variables w, x, y, and z are stored in memory cells 101, 102, 103, and 104, respectively.

|  |  |
| --- | --- |
| Address | Instruction |
| 0 | DECREMENT 104 |
| 1 | LOAD 102 |
| 2 | ADD 103 |
| 3 | SUBTRACT 104 |
| 4 | ADD 101 |
| 5 | STORE 101 |
| 6 | OUT 101 |
| 7 | HALT |

1. (10 pts.) Convert the following assembly language code into Python code. In your answer, the variables stored at locations 101, 102, 103, 104, and 105 should be named a, b, c, d, and e respectively.

|  |  |
| --- | --- |
| 0 | LOAD 103 |
| 1 | ADD 103 |
| 2 | SUBTRACT 102 |
| 3 | STORE 104 |
| 4 | LOAD 105 |
| 5 | COMPARE 104 |
| 6 | JUMPGT 9 |
| 7 | JUMPEQ 9 |
| 8 | JUMPLT 16 |
| 9 | LOAD 101 |
| 10 | ADD 105 |
| 11 | STORE 101 |
| 12 | LOAD 102 |
| 13 | SUBTRACT 103 |
| 14 | STORE 102 |
| 15 | JUMP 19 |
| 16 | LOAD 101 |
| 17 | ADD 105 |
| 18 | STORE 103 |
| 19 | LOAD 101 |
| 20 | ADD 102 |
| 21 | SUBTRACT 103 |
| 22 | STORE 105 |
| 23 | HALT |

1. (4 pts.) Scheme:
   1. What is the result of the following Scheme expression: (2 pts)

(/ 100 (+ 9 (\* 2 2 (sqrt 4)) 3))

* 1. Write a Scheme expression corresponding to the standard mathematical expression: (2 pts)

(6 – 3)/18 + 17/(55 + 11)

Do not evaluate any of the arithmetic operations, i.e. do not convert “6-3” into “3”.

**Part II (50 points) Python Programming**

Your Python code must follow these guidelines. Be aware that if you don’t follow these guidelines, you will lose points, **even if your program runs perfectly**.

* *Always put your name, date, program name, and a general comment describing what your program does at the top of your code. This should be a general statement about how your program behaves.*
* *Variable names should help describe the purpose of a variable.*
* *Function names should help describe the purpose of a function*
* *Every function should be preceded by a comment describing the function’s purpose, input, and output (if any).*
* *Every sound alteration must be done by writing and calling a function. You will lose most of the points for the assignment if you turn in just one big function.*
* *Your functions must communicate properly with your “soundCollage()” main function: do not use any global variables, and any information you need to get from your main “soundCollage()” function to another function must be passed in as a parameter (input) to the function.*
* *The main comment at the top of your program must indicate where you got each of your basic sound files (recorded, Media Sources, Internet, etc.)*

Your assignment is to create a “sound collage”, similar to how you created an image collage for Homework 2. The sound you create must adhere to the following guidelines:

* At least 2 different basic sounds must be used; a rough guideline is to have them be 1-2 seconds long.
* Choose or record your own sounds. You can use something from the internet, media sources, or record something yourself.
* The final sound must be at least 5 seconds long.
* The final sound must have at least 5 “segments”, all separated by 0.3 seconds of silence. (Recall that one simple way to insert silence is to just skip the proper number of samples from the end of one sound until the start of the next.)
* The first 2 segments of your collage must be the original 2 sounds.
* The next 3 segments must be alterations of your original sounds, as follows:
  + Each sound must be altered at least once
  + At least 2 alterations must be from the following list:
    - Alter the frequency enough to be easily audible (adapt Program 116, 117, or 119)
    - Blend the 2 sounds together (adapt Program 110)
    - Echo the sound (adapt Program 111 or 112)
    - Reverse the sound (Program 108)
    - Mirror the sound (Program 109)
    - Set all samples to maximum values (Program 98)
    - Anything else you can think of
  + At least one of the alterations must be from this list:
    - Make the volume increase or decrease audibly 4 or 5 times per second (using just one loop).
    - Reduce or increase the volume gradually throughout the sound
    - Increase or decrease every sample by a different random amount until the sound is just barely recognizable

As in Homework 2, The **soundCollage()** function you write must make all of this happen – all of the above effects and compositing must occur from the single function **soundCollage()**. Of course, I expect you to write and use additional functions to make this easier, but someone (like me!) testing your program must be able to simply call **setMediaPath()**, load your program, then execute **soundCollage()** to hear your collage generated and played.

**NOTE**: A perfectly working program as described above will earn 45 points. To earn the remaining 5 points, you must use your own creativity to go above and beyond those minimal requirements. Examples include additional sound alterations, etc.. Be creative and have fun !!!

**Tips**:

1. Use the template provided on Blackboard as a starting point.
2. All the sounds you use should have the same sampling rate, or things will get weird. Audacity will default to sampling rate 44100. I recommend changing to 22050 before recording so you can use with Media Sources files, which have a sampling rate of 22050.
3. Build on what you’ve learned! If you don’t have your book beside you and your code from the labs in front of you when you’re doing this, then you are going to waste a lot of time. You should also examine and understand the solutions to HW3 as an example of a good way to divide things up into functions and use them.
4. Many of the alterations above have already been written for or by you, in the textbook or in lab, so feel free to make use of that code, but the last required alteration has not been written for you: you need to do that one from scratch, but feel free to get the others done first if that makes it easier for you!

**TURNING IT IN:**

1. Turn in a printout of your code (the hw4.py file) with the rest of your homework.
2. Put a copy of your hw4.py file and a copy of both of your original “.wav” files in your csis110/hw/hw4 folder. THIS IS REQUIRED IN ORDER FOR ME TO RUN IT AND GRADE IT AND FOR YOU TO EARN FULL CREDIT.