CU – FCAI – CS111 – Fundamentals of Computer Sciences 2022/2023

Dr Mohammad El-Ramly©

Part 1: 10 problems to train on machine language for CS Overview Book

1. Write a program to check if a number is divisible by 4 or no and print Y or N. Where ASCII of Y = 59 and of N = 4E. Assume the number is stored at memory location 80_h.
2. Explain why this program has an infinite loop:

```
80  2002
82  2103
84  A103
86  B18C
88  A102
8A  B086
8C  C000
```

3. Translate the following instructions from English into machine language
   - LOAD register 6 with the hexadecimal value
   - LOAD register 8 with the contents of memory cell 8.
   - JUMP to the instruction at memory location C4 if the contents of register 0 equals the value in register B.
   - ROTATE register 4 three bits to the right.
   - AND the contents of registers E and 2 leaving the result in register 1.
4. Write a program in machine language that calculates the sum of odd numbers up to a given number 1 + 3 + 5 +7 + 9 + …. + n
5. Write a machine language program equivalent to the following Python program.

```
i = 1
while i <= 128:
    print (i)
    i = i * 2
```

6. Write a machine language program that computes 4_d x 12_d.
7. Translate this program into machine language program.

```
if x == 0:
    z = 10 + x
else:
    z = 1 + x
```

   Assuming that x and z are stored at bytes C0 and C1 and the program starts at address 80.
8. Write a program in machine language that can handle a negative number in 2's complement format and extracts the absolute value of it.
9. Write a machine language program that can calculate the multiplication table of 2 (1 x 2 to 12 x 2) and stores it in 12 memory locations. Do not hardcode values. Uuse loops.
10. Write a machine language program that loops over a range of memory locations and multiplies each one by 2 and stores it back. It stops if it finds a –ve number.

Vole machine:

Machine has 256 memory locations; each one is one byte. Memory addresses start from 00 to FF. It has 16 registers; each is 1-byte width. First register is 0 and last one is F. It has a 1-byte program counter to hold the next instruction. It also has a 2-bytes instruction register that holds the next instruction for decoding after it is fetched from memory. It supports the following instruction set.

Machine Language

Op-code	Operand	Description
1	RXY	LOAD the register R with the bit pattern found in the memory cell whose address is XY. *Example:* 14A3 would cause the contents of the memory cell located at address A3 to be placed in register 4.
2	RXY	LOAD the register R with the bit pattern XY. *Example:* 20A3 would cause the value A3 to be placed in register 0.
3	RXY	STORE the bit pattern found in register R in the memory cell whose address is XY. *Example:* 35B1 would cause the contents of register 5 to be placed in the memory cell whose address is B1.
3	**R00**	**STORE to location 00, which is a memory mapping for the screen. Writing to 00 is writing to screen.**
4	0RS	MOVE the bit pattern found in register R to register S. *Example:* 40A4 would cause the contents of register A to be copied into register 4.
5	RST	ADD the bit patterns in registers S and T as though they were two's complement representations and leave the result in register R. *Example:* 5726 would cause the binary values in registers 2 and 6 to be added and the sum placed in register 7.
6	RST	ADD the bit patterns in registers S and T as though they represented values in floating-point notation and leave the floating-point result in register R. *Example:* 634E would cause the values in registers 4 and E to be added as floating-point values and the result to be placed in register 3.
7	RST	OR the bit patterns in registers S and T and place the result in register R. *Example:* 7CB4 would cause the result of ORing the contents of registers B and 4 to be placed in register C.
8	RST	AND the bit patterns in register S and T and place the result in register R. *Example:* 8045 would cause the result of ANDing the contents of registers 4 and 5 to be placed in register 0.
9	RST	EXCLUSIVE OR the bit patterns in registers S and T and place the result in register R. *Example:* 95F3 would cause the result of EXCLUSIVE ORing the contents of registers F and 3 to be placed in register 5.
A	R0X	ROTATE the bit pattern in register R one bit to the right X times. Each time place the bit that started at the low-order end at the high-order end. *Example:* A403 would cause the contents of register 4 to be rotated 3 bits to the right in a circular fashion.
B	RXY	JUMP to the instruction located in the memory cell at address XY if the bit pattern in register R is equal to the bit pattern in register number 0. Otherwise, continue with the normal sequence of execution. (The jump is implemented by copying XY into the program counter during the execute phase.) *Example:* B43C would first compare the contents of register 4 with the contents of register 0. If the two were equal, the pattern 3C would be placed in the program counter so that the next instruction executed would be the one located at that memory address. Otherwise, nothing would be done and program execution would continue in its normal sequence.
C	000	HALT execution. *Example:* C000 would cause program execution to stop.

Part 1: 7 problems to train on Python

11. Encryption is very important to protect sensitive data. One simple way of encryption is to replace each letter with another letter. Write a function that reads from a file a list of pairs for letters; the first is the original and the second is the one to replace it. It should do this for English language letters and important symbols. Then it uses this dictionary for encryption or decryption of messages.

    For example, the file may have pairs like this when loaded to dictionary: {'a' : 'n', 'b' : 'f', ' ' : '.', ……} . Then the program will load the file and ask the user if he wants to encrypt or decrypt a message. The user will enter a message and the program will perform the desired action on it.

12. Develop a Python program that allows the user to enter student IDs, names and marks in three subjects out of 100. The program allows the user to get the max and min mark in each subject, to calculate the GPA of a given student out of 4 (GPA = average of three marks / 25), get the average mark in a given subject. Decide on the suitable data structures to use.

13. Modify the previous program so that it loads the students' marks from a file.

14. Develop two recursive functions that calculates m^n, one according to each of the following definitions:

$m^n = \begin{cases} 1 & \text{if } n == 0 \\ m \times m^{n-1} & \text{if } n > 0 \end{cases}$	$m^n = \begin{cases} 1 & \text{if } n == 0 \\ m^{n//2} \times m^{n//2} \times m & \text{if } n > 0 \text{ and n is odd} \\ m^{n/2} \times m^{n/2} & \text{if } n > 0 \text{ and n is even} \end{cases}$

    What is the complexity of each function? Which one is faster? How many times, each function will be called, if we want to calculate m^{20}?

15. Python has a data structure called Set that is a container of elements that (1) does not allow duplicate items and (2) does not have order for elements and does not allow indexing. Go to Python interpreter and type:

```
>>> set1 = set ([1,2,3,4,4])
>>> set2 = set ([3,4,5,6,6])
>>> dir (set1)
```

    See what functions are available for Set data structure. Apply some of these functions and see the results.

    Then write a program that loops forever and give the user the options to (1) enter new sets, (2) calculate their union, (3) calculate their intersection, (4) calculate Set1 – Set2 and Set2 – Set1, (5) or quit.

16. Assume Python has no Set data structure. Implement the program in the previous problem using lists and list comprehension. Note that when creating a set, you need to reject redundant items. Implement the same six functions above.

17. Write a program that takes a word and a file name and return the number of occurrences of this word in the file.

18. Modify the program to find the word even if letters where in a different case and write the line numbers where the word exists.