CSI 333 – Programming at the Hardware-Software Interface  
SQUPT, Spring 2019

**Project 5**

The total grade for the assignment is 100 points.

You must follow the programming and documentation guidelines (see file *Programming Assignments Requirements and Recommendations.docx*).

This is a **team project** (except for those students who have opted to work on their own). Group of **two** students may work on this project together.

**Due date: 11:59pm Sunday, May 19, 2019**

# Description

You are required to write a MAL program that prompts a user for a line of characters (only decimal digits or whitespace characters) and reads the line typed by the user. If the line contains just whitespace characters your program should simply output the message “Line contains only whitespace characters.” and stop. Otherwise, your program should compute and output the following:

1. The total number of *decimal* *integers* (not digits) in the input line.
2. The maximum number of 1's in the binary representations of the integers in the input line.
3. All given integers with the maximum number of 1's in the binary representations. The integers should be printed in order as they appear in the input line.

Example:

Suppose the user types:

12 346 31 44 73

Then,

|  |  |  |
| --- | --- | --- |
| Integers | Binary representation | Number of 1’s |
| 12 | 1100 | 2 |
| 346 | 101011010 | 5 |
| 31 | 11111 | 5 |
| 44 | 101100 | 3 |
| 73 | 1001001 | 3 |

For this example, the required answers are as follows:

1. The number of the given integers is 5.
2. The maximum number of 1's in the binary representations of the given integers is 5.
3. The given integers that have four 1's in the binary representations are 346 and 31.

Program outline:

The outline for your program must be the following.

1. Prompt the user for a line of characters.
2. Read the line of characters typed by the user.
   1. If the line has only whitespace characters print the message:   
      "Line contains only white space characters." and stop.
   2. Otherwise compute the quantities mentioned above and print the answers.
3. Stop.

Assumptions:

* The characters typed by the user are decimal digits or whitespace character. Hence, there is no minus sign and there are no negative numbers in the input line.
* A whitespace character refers to a space, a tab or the newline character.
* The user does not exceed 9 decimal digits sequentially, so four bytes are enough for any integer in the input.
* Any line typed by a user has at most 80 characters including the newline character.
* End of any line is determined by the null character.
* Each time your program is executed, it should handle just one line of characters.

Programming Suggestions:

1. MAL program must be in a file named p5.asm.
2. It must have at least two functions in addition to the main program.
3. Study materials of *Class 12-Data structures in MAL* before working.
4. Check whether the input line consists of just whitespace characters, no other error checks are needed.
5. There is no need to convert the integer to binary; when the integer is in the register, it is already in binary form. Use bitwise operations to count the number of 1’s.

# Example of program execution and sample data to test your program

Important Note: Some sample inputs that can be used to test your programs are given below. However, you should remember that when we compile and run your source files, we will use other data. Just because your programs work for the sample inputs given below, you shouldn't assume that they will work for all inputs. Therefore, you should test your programs thoroughly with other input values.

Example 1 of program execution:

Enter the line? 12 346 31 44 73

No. of integers: 5

Maximum number of 1’s in the binary representation: 5

Integers of maximum number of 1’s in the binary representation:

346

31

Example 2 of program execution:

Enter the line? 27 53 89 3 10 111 233

No. of integers: 7

Maximum integer: 233

Maximum number of 1’s in the binary representation: 6

Integers of maximum number of 1’s in the binary representation:

111

Example 3 of program execution:

Enter the line? 73 15 786 256 90 266

No. of integers: 6

Maximum integer: 786

Maximum number of 1’s in the binary representation: 4

Integers of maximum number of 1’s in the binary representation:

15  
786  
90

# Submission

You must perform submissions as directed by your co-instructor.

*Ignoring any of the following rules will result in penalty or even ZERO grade for the project!*

For the **team project** each team must make **only one** submission. That is, in each team, ONLY ONE member must do this. Team submissions must include additional documentation in the source file as explained below.

Submission should include:

* A file named as directed by your co-instructor with source code for the project. More details will be given in your lab classes.
* Screenshot with program output.

At the top of your source code file the following information must appear in the form of comments:

* 1. Course code and title (i.e. “CSI 333. Programming at the Hardware-Software Interface”),
  2. Semester (e.g., Spring 2019),
  3. The name of your lab classes supervisor,
  4. Your class (e.g., ZR170102),
  5. Your student ID,
  6. Your pinyin name.

Students working in a team must have the following information at the beginning of your source file in the form of comments:

* 1. Course code and title (i.e. “CSI 333-Programming at the Hardware-Software Interface”),
  2. Semester (e.g., Spring 2019),
  3. The name of your lab classes supervisor,
  4. Your class (e.g., ZR170102),
  5. The student IDs of the two team members,
  6. The pinyin names of the two team members,
  7. A clear explanation of how the work for the project was divided among the two team members. Indicate clearly who developed each function and how the testing work was divided between the team members.

Make sure that your program produces correct results on MARS simulator. Programs that cause errors will NOT receive any credit.

# Project Grading

Programs will be graded by co- instructors.

For students working individually:

1. Correctness: 85 points
2. Structure and documentation: 15 points

For students working in a team:

1. Correctness: 65 points
2. Structure and documentation: 15 points
3. Team work: 20 points

Each team member must participate in developing, documenting and testing the program. Each team should include additional documentation at the beginning of the source file indicating how the work for the project was divided between the two team members. (Indicate clearly who developed each function and how the testing work was divided between the team members.) After the submission deadline, each team must meet with their instructor who supervises the lab classes. During the meeting, the instructor will ask questions about the team’s program and determine the points for team work. (The two team members may receive different scores for team work.)

# Additional comments

Comment 1:

You may study and reuse some ideas of the following code. *Important*: it is NOT a solution for your project, but it has some useful hints.

#this code identifies the number of whitespace, or numbers of integers,

#by assuming that every integers are separated by one space, tab, or newline

.data

prompt: .asciiz "Enter Numbers: \n"

str: .space 80

.text

main:

la $a0,prompt #print out the prompt

li $v0,4

syscall

la $a0,str #read the input str with 80 space

li $a1,80

li $v0,8

syscall

la $a0,str #print out the str

li $v0,4

syscall

move $s1,$0

la $a0,str

move $t0,$a0 #move str to $t0

loop:

addi $t0,$t0,1

lb $t1,0($t0) #load first char of str into $t1 to be analyzed

beqz $t1,exit

beq $t1,32,count #if ' ' branch

beq $t1,11,count #if '\t' branch

beq $t1,10,count #if '\n' branch

j loop

count:

addi $s1,$s1,1

j loop

exit:

move $a0,$s1

li $v0,1

syscall

li $v0,10 #exit

syscall

Comment 2:

Project 5 has many details that you need to take care of. Below there is a *pseudo code* (or an algorithm) to guide you in your coding. You may use your own idea to build the project, this is only a suggestion.

*main segment*

*data*

*input: array of bytes of size 80*

*//e.g. input 15, 32, 0, 12, 18 as array of characters*

*integers: array of words of size 40*

*//**will save the actual values of integers*

* 1. *input <- read*
  2. *check empty line   
     //subroutine 1, to check if input is all white-spaces, if so, the program terminates*
  3. *construct integers   
     //subroutine 2, to convert ASCII code representation of digits to actual integers, 49 is the ASCII code of '1', 53 is for '5', so 15 will be saved as word instead of 49 and 53 as two bytes.*
  4. *print number of integers   
     //subroutine 3, to count and print number of integers, here is 5*
  5. *print the maximum number of '1' bits   
     //subroutine 4*
  6. *print integers with max. number of '1' bits   
     //subroutine 5*
  7. *end and stop*

*subroutine 1: check empty line*

* 1. *load the byte from input as C*
  2. *if C is the end of the line then print "Empty line", end and stop*
  3. *if C is not a digit, goto step 1*
  4. *return to main*

*subroutine 2: construct integers*

*// read handout 15.3*

* 1. *load the first/next byte from input as C*
  2. *if C is the end of the line, goto step 19*
  3. *if C is not a digit goto step 1*
  4. *E = 0   
     //to count number of digits of an integer number*
  5. *C = C - 48   
     //48 is the ASCII code for '0', the subtraction is to get the actual value*
  6. *E = E + 1*
  7. *push C*
  8. *load the next byte from input as C*
  9. *if C is a digit then goto step 5*
  10. *sum = 0, P = 1, R = 10   
      //P is to control digit's placement, ones, tens, ...*
  11. *if E == 0, goto step 17*
  12. *E = E - 1*
  13. *pop D   
      //get the digit from stack*
  14. *sum = sum + D \* P*
  15. *P = P \* R*
  16. *goto step 11*
  17. *save sum in integers*
  18. *goto step 1   
      //get ready for the digits of the next integer*
  19. *save -1 in integers   
      //indicating the end of numbers, will be useful in the coming subroutines*
  20. *return to main*

The remaining subroutines are your responsibility.

Good luck!

**1 1 3**

**1 0 1 1 11**

**1 1**