國立清華大學資訊工程學系
11010 CS 410000 計算機結構
Homework 2

Deadline: 2021/10/25 (23:59)

There are two parts in this homework.

PART I. (Load, Store, Add, Sub)

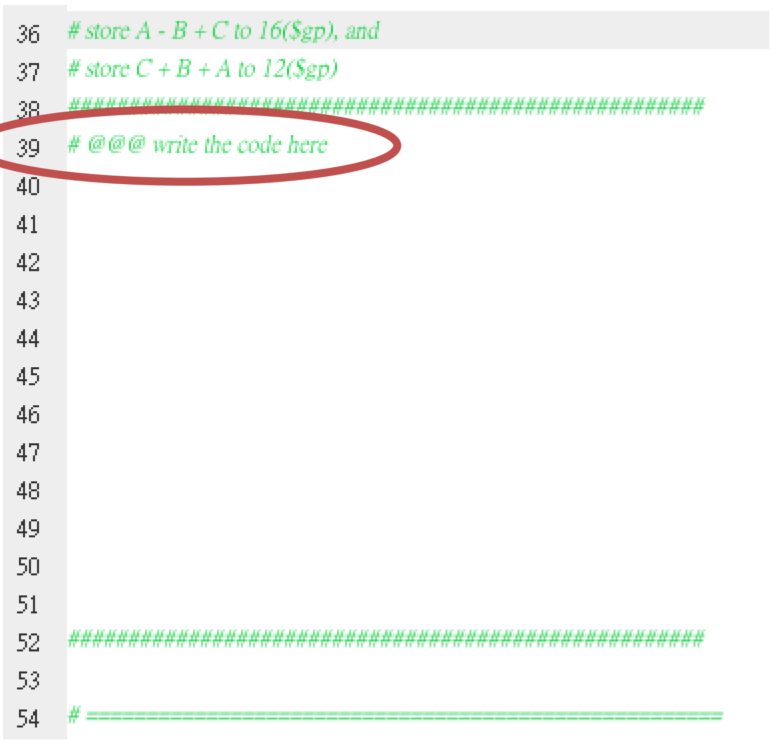
Please load the data 8($gp) as A, 4($gp) as B, and 16($gp) as C, and do the following calculations.

D = A - B + C, store D to 16($gp)

E = C + B + A, store E to 12($gp)

Hint

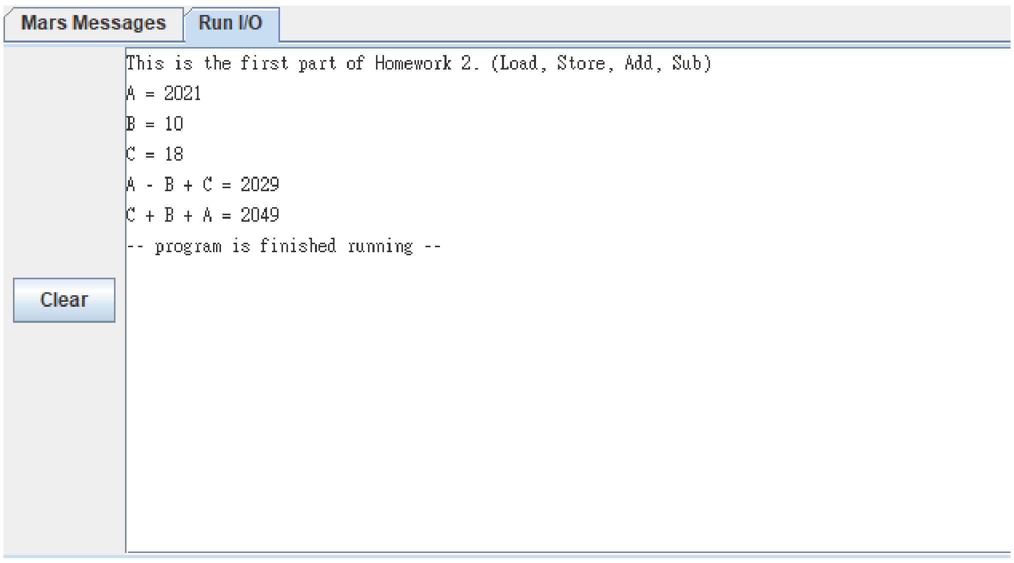
We will give a template called **arch_hw2_p1_template.asm**, just open it using Mars4_5.jar, write your code within the ####### block in the file (i.e., line 36~41), but **DO NOT** modify the code elsewhere. Please refer to the following figure.



```
36  # store A - B + C to 16($gp), and
37  # store C + B + A to 12($gp)
38  ####################################################
39  # @@@ write the code here
40
41
42
43
44
45
46
47
48
49
50
51
52  ####################################################
53
54  # ===============================================
```

After you write your code, save it.

Next, press “F3” to assemble the code. (Make sure there is no error!)
Next, press “F5” to run.
The “Run I/O” screen should show the result like the following figure.



```
This is the first part of Homework 2. (Load, Store, Add, Sub)
A = 2021
B = 10
C = 18
A - B + C = 2029
C + B + A = 2049
-- program is finished running --
```

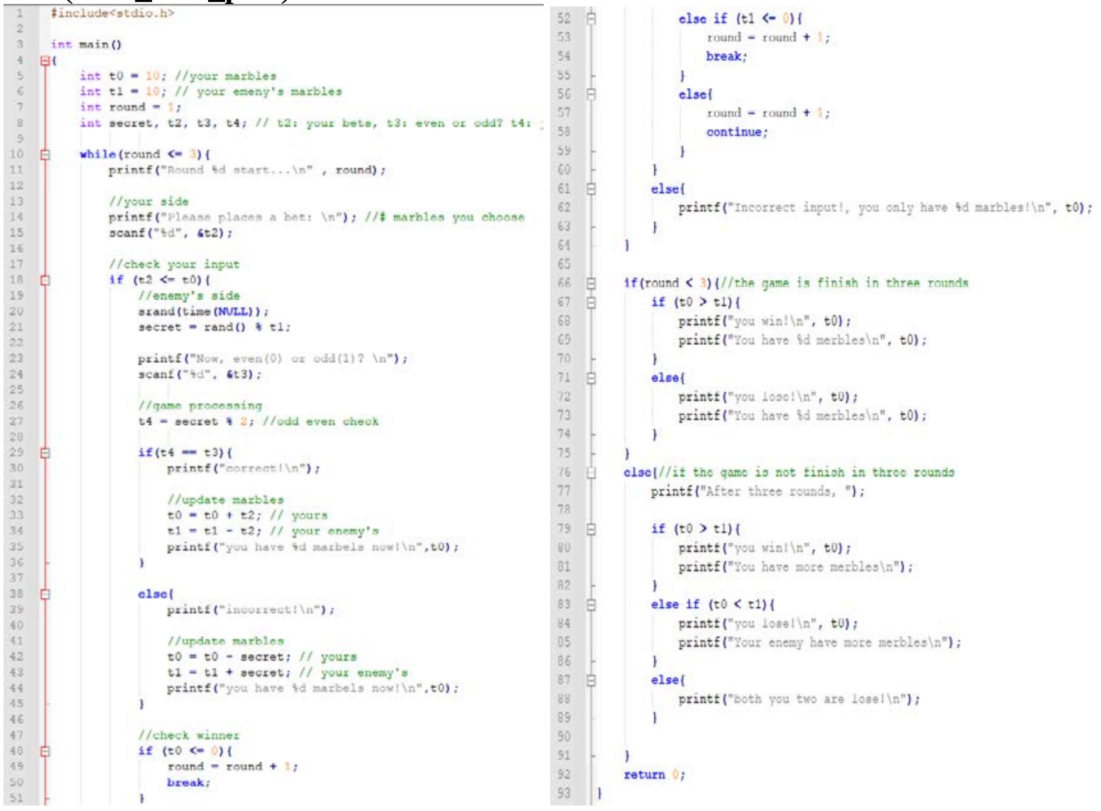
PART II. (Branch Loop, System call, Arithmetic Operations)

Please convert the following C-like code to MIPS assembly code. Write a new assembly file for this part.

Description: Please design a “Marbles game” by MIPS.

It has only three rounds. Each of you has 10 marbles. In each round, you have to enter a bet number and guess your enemy’s bets (even or odd). Once you have correct guessing, your enemy needs to give you his marbles (according to your bets). On the other side, you have to give yours (according to his bets) if you have incorrect guessing. The game will stop when one of you has no marbles or three rounds later. Whoever has the most marbles is the winner.

Here is a screenshot of the C code, you can find the completed C code in the attached files (**arch_hw2_p2.c**)



```c
#include<stdio.h>

int main()
{
    int t0 = 10; //your marbles
    int t1 = 10; // your emeny's marbles
    int round = 1;
    int secret, t2, t3, t4; // t2: your bets, t3: even or odd? t4:

    while(round <= 3){
        printf("Round %d start...\n" , round);

        //your side
        printf("Please places a bet: \n"); //# marbles you choose
        scanf("%d", &t2);

        //check your input
        if (t2 <= t0){
            //enemy's side
            srand(time(NULL));
            secret = rand() % t1;

            printf("Now, even(0) or odd(1)? \n");
            scanf("%d", &t3);

            //game processing
            t4 = secret % 2; //odd even check

            if(t4 == t3){
                printf("correct!\n");

                //update marbles
                t0 = t0 + t2; // yours
                t1 = t1 - t2; // your enemy's
                printf("you have %d marbels now!\n",t0);
            }

            else{
                printf("incorrect!\n");

                //update marbles
                t0 = t0 - secret; // yours
                t1 = t1 + secret; // your enemy's
                printf("you have %d marbels now!\n",t0);
            }

            //check winner
            if (t0 <= 0){
                round = round + 1;
                break;
            }
            else if (t1 <= 0){
                round = round + 1;
                break;
            }
            else{
                round = round + 1;
                continue;
            }
        }
        else{
            printf("Incorrect input!, you only have %d marbles!\n", t0);
        }
    }

    if(round < 3){//the game is finish in three rounds
        if (t0 > t1){
            printf("you win!\n", t0);
            printf("You have %d merbles\n", t0);
        }
        else{
            printf("you lose!\n", t0);
            printf("You have %d merbles\n", t0);
        }
    }
    else{//if the game is not finish in three rounds
        printf("After three rounds, ");

        if (t0 > t1){
            printf("you win!\n", t0);
            printf("You have more merbles\n");
        }
        else if (t0 < t1){
            printf("you lose!\n", t0);
            printf("Your enemy have more merbles\n");
        }
        else{
            printf("both you two are lose!\n");
        }

    }
    return 0;
}
```

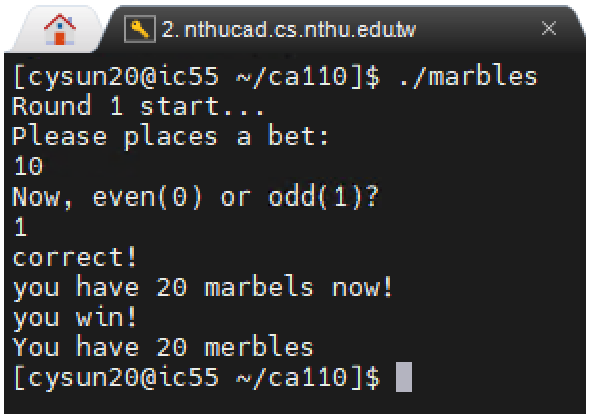
Hint

a You can refer to the template in Part I or Appendix to learn how to do `printf` and `scanf` in MIPS.

b Two references for finding the functionality of MIPS instruction.
(English: http://alumni.cs.ucr.edu/~vladimir/cs161/mips.html)
(中文: https://blog.xuite.net/tzeng015/twblog/113272086-MIPS+%E6%8C%87%E4%BB%A4%E9%9B%86)

執行畫面(補充)說明:

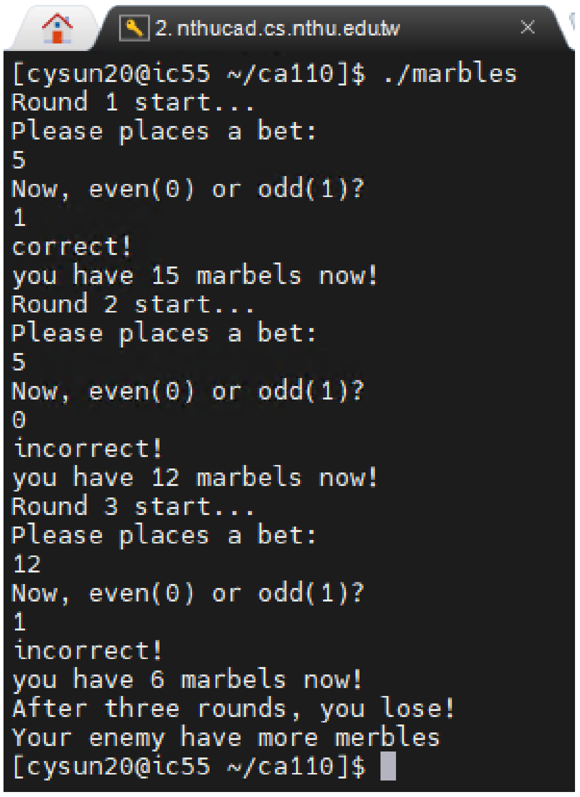
Example 1:



```
[cysun20@ic55 ~/ca110]$ ./marbles
Round 1 start...
Please places a bet:
10
Now, even(0) or odd(1)?
1
correct!
you have 20 marbels now!
you win!
You have 20 merbles
[cysun20@ic55 ~/ca110]$
```

一開始你跟對手都有10顆彈珠
在第一回合你下注10顆，猜對方為奇數
你猜中了因此對方要給你10顆
你遊戲贏了，對方沒有彈珠了

Example 2:



```
[cysun20@ic55 ~/ca110]$ ./marbles
Round 1 start...
Please places a bet:
5
Now, even(0) or odd(1)?
1
correct!
you have 15 marbels now!
Round 2 start...
Please places a bet:
5
Now, even(0) or odd(1)?
0
incorrect!
you have 12 marbels now!
Round 3 start...
Please places a bet:
12
Now, even(0) or odd(1)?
1
incorrect!
you have 6 marbels now!
After three rounds, you lose!
Your enemy have more merbles
[cysun20@ic55 ~/ca110]$
```

在第一回合你下注5顆，猜對方為奇數
你猜中了因此對方要給你5顆 (此時你有15顆)
第二回合中你一樣下注5顆，猜對方為偶數
你猜錯了因此要給對方3顆 (3顆是因為對方下注3)
第三回合中你下注了手上全部的彈珠，猜對方為奇數
你猜錯了因此要給對方6顆 (6顆是因為對方下注6)
第三回合結束後，你與對手都還有彈珠，此時比誰的彈珠多
你剩下6顆，對手14，對手贏

Hint

a. We will give the C code called **arch_hw2_p2.c** for reference. We will also give a MIPS template called **arch_hw2_p2_template.asm**. We strongly recommend you do this part by yourself, or you can refer to the template if you need some help.

b. TA will use other numbers to test if your program is correct.

Submission (Two assembly programs)

Please name your assembly program with your student ID; for example, **arch_hw2_p1_102062801.asm** & **arch_hw2_p2_102062801.asm**, and upload these 2 files onto iLMS. (https://eeclass.nthu.edu.tw/course/3255)

Grading Criteria

Correctness: 80%

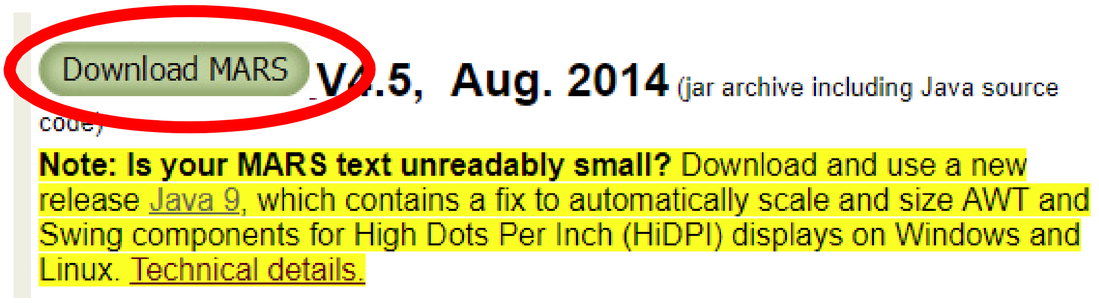
Comments in your code: 10%

Output format: 10%

嚴格禁止抄襲，抄襲者與被抄襲者一律 0 分！

MARS (MIPS Assembler and Runtime Simulator)

1. MARS can assemble and simulate the execution of MIPS assembly language programs. Please refer to the following URL to download Mars4_5.jar: http://courses.missouristate.edu/kenvollmar/mars/download.htm

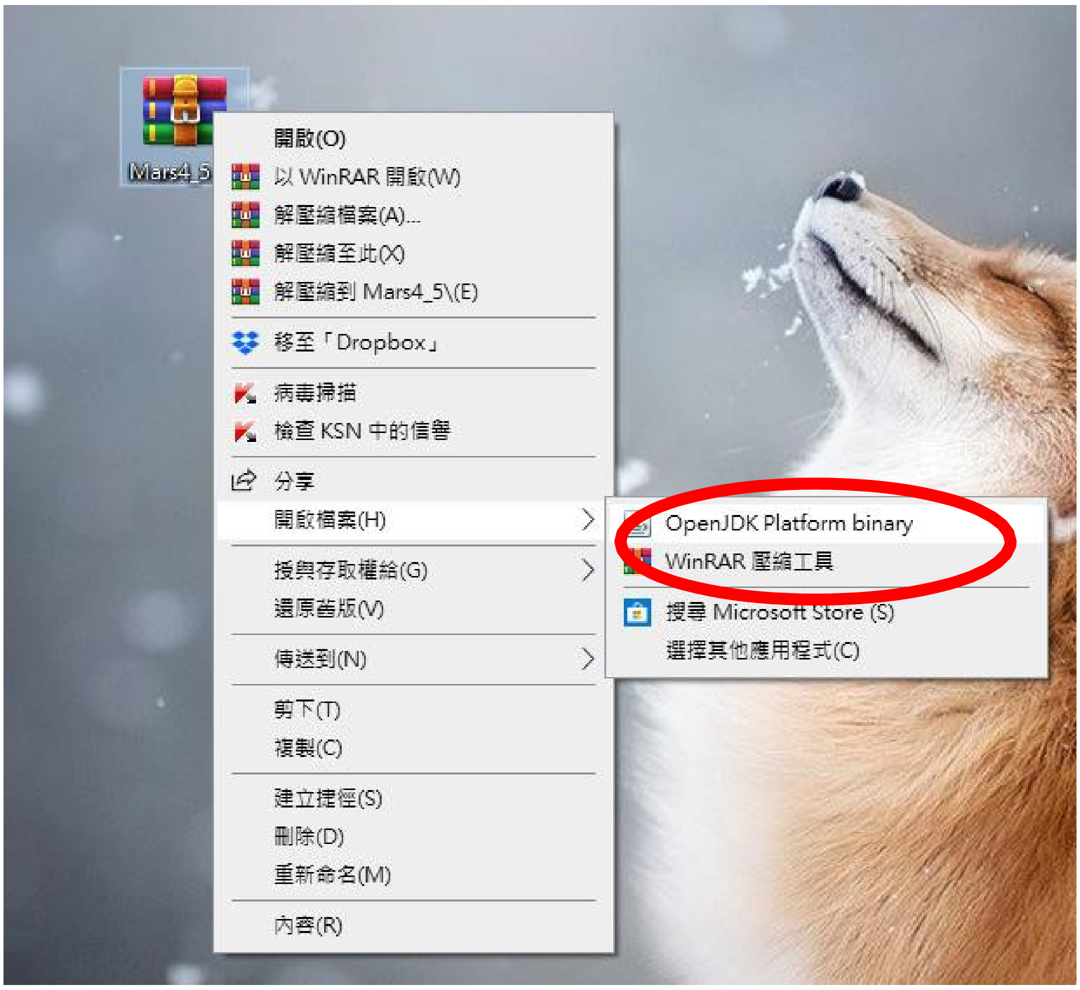


Download MARS V4.5, Aug. 2014 (jar archive including Java source code)

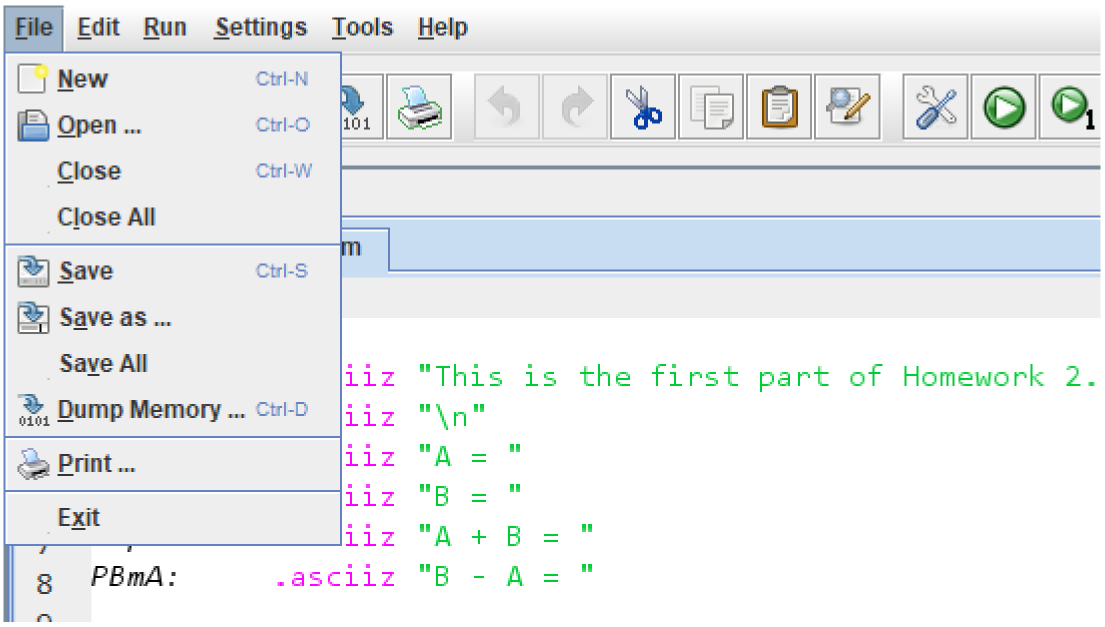
Note: Is your MARS text unreadably small? Download and use a new release Java 9, which contains a fix to automatically scale and size AWT and Swing components for High Dots Per Inch (HiDPI) displays on Windows and Linux. Technical details.

2. MARS is developed with Java language, and it requires JRE (Java Runtime Environment) installed on your computer. Please refer to the following URL to download JRE: https://www.oracle.com/java/technologies/downloads/

3. After you download the MARS, it is a ".jar" file. Please **DO NOT** decompress it. You can open the MARS by following method.

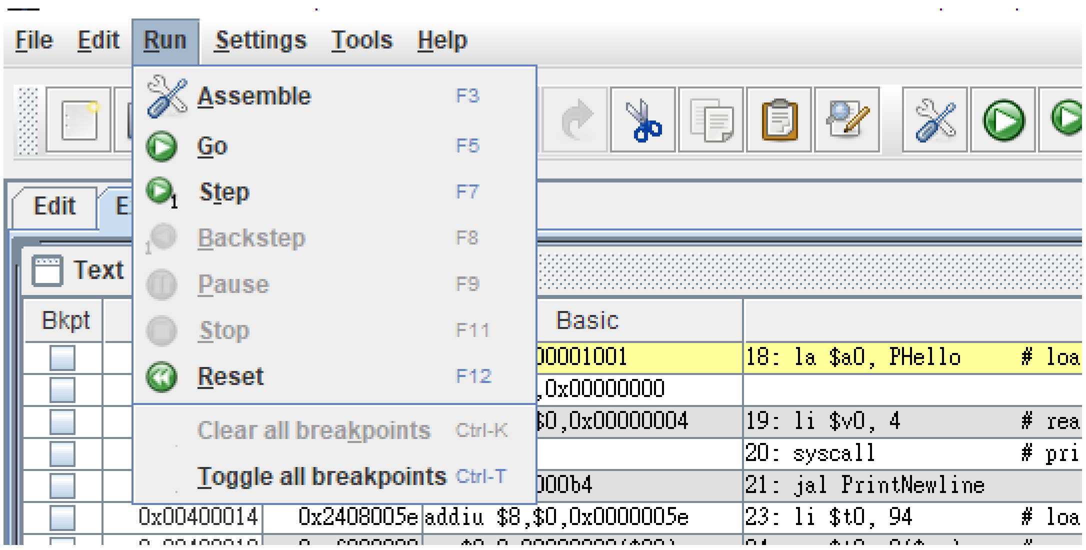


4. Usage of MARS:



```
iiz  "This is the first part of Homework 2.
iiz  "\n"
iiz  "A = "
iiz  "B = "
iiz  "A + B = "
PBmA:       .asciiz  "B - A = "
```

(a) New, Open, Save and Close



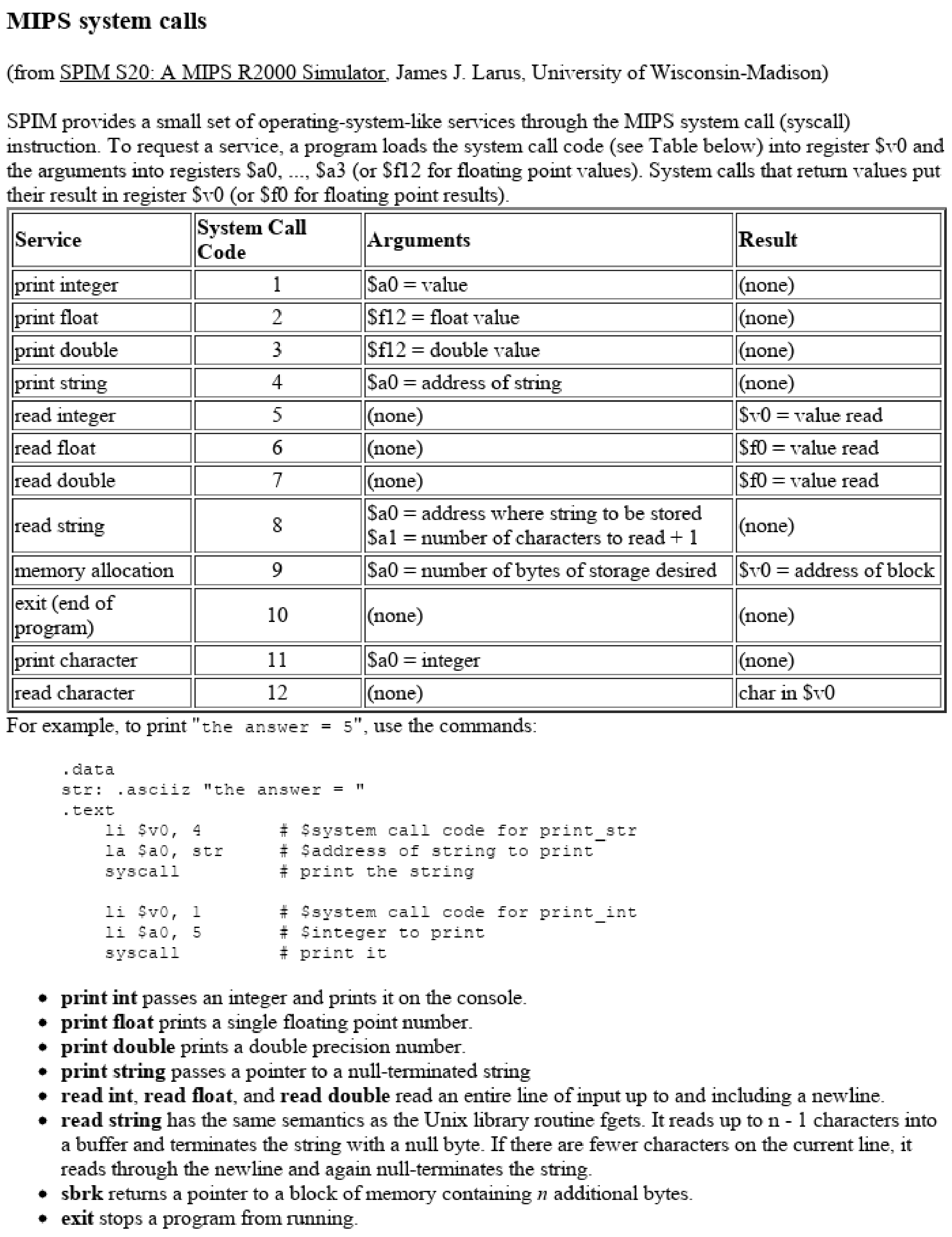
```
                          00001001                 18: la $a0, PHello      # loa
                          ,0x00000000
                          $0,0x00000004            19: li $v0, 4           # rea
                                                   20: syscall             # pri
                          000b4                    21: jal PrintNewline
0x00400014  0x2408005e  addiu $8,$0,0x0000005e     23: li $t0, 94          # loa
```

(b) Assemble and then Go (Run)

P. S. Save your file, Assemble and Go

Appendix

(Source: http://students.cs.tamu.edu/tanzir/csce350/reference/syscalls.html)



MIPS system calls

(from SPIM S20: A MIPS R2000 Simulator, James J. Larus, University of Wisconsin-Madison)

SPIM provides a small set of operating-system-like services through the MIPS system call (syscall) instruction. To request a service, a program loads the system call code (see Table below) into register $v0 and the arguments into registers $a0, ..., $a3 (or $f12 for floating point values). System calls that return values put their result in register $v0 (or $f0 for floating point results).

Service	**System Call Code**	**Arguments**	**Result**
print integer	1	$a0 = value	(none)
print float	2	$f12 = float value	(none)
print double	3	$f12 = double value	(none)
print string	4	$a0 = address of string	(none)
read integer	5	(none)	$v0 = value read
read float	6	(none)	$f0 = value read
read double	7	(none)	$f0 = value read
read string	8	$a0 = address where string to be stored $a1 = number of characters to read + 1	(none)
memory allocation	9	$a0 = number of bytes of storage desired	$v0 = address of block
exit (end of program)	10	(none)	(none)
print character	11	$a0 = integer	(none)
read character	12	(none)	char in $v0

For example, to print "`the answer = 5`", use the commands:

```
.data
str: .asciiz "the answer = "
.text
    li $v0, 4        # $system call code for print_str
    la $a0, str      # $address of string to print
    syscall          # print the string

    li $v0, 1        # $system call code for print_int
    li $a0, 5        # $integer to print
    syscall          # print it
```

- **print int** passes an integer and prints it on the console.
- **print float** prints a single floating point number.
- **print double** prints a double precision number.
- **print string** passes a pointer to a null-terminated string
- **read int**, **read float**, and **read double** read an entire line of input up to and including a newline.
- **read string** has the same semantics as the Unix library routine fgets. It reads up to n - 1 characters into a buffer and terminates the string with a null byte. If there are fewer characters on the current line, it reads through the newline and again null-terminates the string.
- **sbrk** returns a pointer to a block of memory containing *n* additional bytes.
- **exit** stops a program from running.