CS 354 - FALL 1994
Assignment 9

Grader: Gary Zipfel
Due before noon on 11/16/94

1 Purpose

The purpose of this assignment is to write a *MAL* program that deals with arrays. It is the same program as in assignment seven, only now it must be written in the *MAL* programming language. In writing this program you will become familiar with the **MIPS** register usage conventions.

2 The Problem

You are to implement a *random walk* , forward or backward through an array of 8 elements. The user enters a *3 digit* positive seed for random number generation. Using the seed, a random number n is generated scaled to

```
[-2 -1 0 +1 +2 ]
```

Walk forward or backward n steps depending on n being positive or negative.

Keep a running total within each element, each time adding in the random number that takes you to that element. If the walk takes you out of the array in either direction you should go to the last element of the array (ie.,*a[7]*). Add the integer that attempted to take you out of the array to the last element and continue.

At the end of 20 iterations print out the contents of each of the array elements and stop. The array elements are to be initialized with zero and you are at the zeroth element to start with.

3 Sample Run

Several sample runs of the program might look (Note, the sequence given is an example, and may not be the sequence generated by the actual code):

```
Enter the seed for random number generation : 383

Sequence of random numbers generated

2 -1 2 1 -1 0 2 1 1 2 1 -1 2 -1 0 0 -1 -2 -1 1

Contents of the array :
0  0
1 -1
2  1
3  0
4  1
5  1
6 -1
7  6
```

4 Requirements

The following requirements must be followed to receive full credit for the assignment. Read these carefully, and make sure that you have satisfied all of them.

- You may assume that the integer entered by the user will be small enough that it will not cause an overflow. This does not mean that the input will be perfect, only that the numeric value of the number entered will not be larger than the largest value which the machine can handle.
- If your program encounters illegal input of any kind, it should print an error message followed by a linefeed and halt.
- In order to simplify debugging, and to better enforce the use of **MIPS** register usage conventions, **you are restricted to using registers t0 - t9 , and a0 - a3. Registers s0 - s8 are not to be used.** Note: If an instruction expects you to use another register to pass a parameter (ie/ the *puts* instruction) you must use it. The restriction only applies to the code which you write.
- The program must contain procedures. You will be given a procedure called *get_rand* which is passed a seed, and returns an updated seed (to be used in the next call to get_rand) and the random number generated. The input/output details of this procedure will be given in the header comments of the file itself. The get_rand procedure is located in the public directory of cs354-1, and is in the file called get_rand.mal.
- You must also create procedures called *print_integer* , which takes an integer value in a register and prints it's value to the screen using *putc* , and *print_array* which prints the values of each array slot. The *print_integer* procedure will be used to print the random numbers generated by *get_rand* and to print the final array values in *print_integer* . The *print_integer* procedure should expect the input integer (the number to be printed out) to be in register a0.

  Note: This is very much like assignment 4's output routine.
- Your main section must have the following format:
  - *Get_seed* : a procedure which reads in the random number seed initially.
  - Loop:
    - Call to procedure *get_rand* : This will also call *print_integer* to print the random number generated.
    - Update of the array (can be a procedure call, but doesn't have to be).
    - Check loop boundaries (20 iterations).
  - Call to your procedure *print_array* : *print_array* prints the elements of the array. It must call *print_integer* .

5 How to turn in your assignments

You are to turn in the source code for your program by using the *handin9* program. When you have completed the assignment, hand in your source code by typing:

```
~cs354-1/bin/handin9 filename
```

where filename is the file containing the source code. You will receive feedback from the grader of your assignment via a grade sheet. DO NOT hand in a hard copy of the source code. An email message containing the file you send in via handin9 will be sent to you. Keep this file in a safe place for future reference. It will be a uuencoded version of your source code.