CS 232 – Computer Architecture I
Rose-Hulman Institute of Technology
Computer Science and Software Engineering Department

Lab 4 - MIPS Procedures II

1 Objectives

Following completion of this lab you should be able to:

- Call and pass arguments to procedures in MIPS programs.
- Return from procedures.
- Save and restore registers.
- Correctly manage the stack.
- Discuss register allocation strategies.

2 General

Be sure to follow the MIPS procedure calling conventions as described in your book (Section 3.6 and Appendix A.6).

The required output format is mandatory.

3 Sort Static Array

`sort.asm` contains the main function of a program to sort an array. It also contains a couple of helper procedures – *shift* and *print*. It calls the procedure *sort* that you will write which in turn should calls the procedure *shift*. Make sure that you understand the procedure calling convention implemented in this code.

Modify `sort.asm` by adding a procedure called *sort* which implements insertion sort. **Do not** modify the code for *main*, *shift* or *print*. *sort* should implement the algorithm shown in the following C code:

```
void
sort(int n, int V[]) {
  int i, j, key;

  for (j=1; j<n; j++) {
    key = V[j];
    i = j-1;
```

```
      while ((i >= 0) && (V[i] > key)) {
        shift(i, V);
        i--;
      }
      V[i+1] = key;
    }
  }.
```

V is a global array, but is passed as an argument to *sort*. Make sure you understand the algorithm. As part of the comment block for you procedure be sure to document the register allocation. Test you program using several different arrays.

4 Sort a Dynamic Array

Copy the code in file `sort.asm` to `dsort.asm`. Modify `dsort.asm` to:

1. Prompt the user for the size k of the input array with the string "How many numbers?". You may choose to restrict the range of k, however do it correctly and document it.
2. Dynamically allocate an array of size k in memory using the `SPIM` system call *sbrk*[1]. This system call expects an integer i in register \$a0 and returns a pointer to a block of i **bytes** of memory in register \$v0.
3. Prompt the user for k integers and store them in the dynamically allocated array.
4. Print the input array.
5. Sort the array and print it.

Be sure to thoroughly test your program. Executing the program should look like this:

```
How many numbers? 5
Input: 1
Input: 2
Input: 3
Input: 4
Input: 5
The input array: 1 2 3 4 5
The sorted array: 1 2 3 4 5
```

5 Peer review

Prior to submitting your lab, get another group to look at your *sort* procedure. Each group should review **exactly** one other groups procedure. The reviewing group should submit a brief written

[1]See page A-49 of your text for more information.

statement about whether or not the procedure follows the `MIPS` procedure calling conventions. Also include some general comments about the strengths and weakness of the procedure as coded.

6 Turning It In

When the lab is complete (not later than 12 December 2001), upload all of your files (`sort.asm` and `dsort.asm` to:

`/Class/cs/cs232/turnin/`<*section*>`/`<*username*>`/lab4/`.