AUTUMN 2016 CSE 2421 LAB 4

DUE: Sunday, November 13, by 11:30:00 p.m.

Objectives:

- Assembly language programs in Y86
- Using assembler directives to initialize RAM for static input data, output, and the stack in Y86
- Move, ALU, jump, call, and return instructions in Y86
- Stack management in Y86

REMINDERS:

– You are allowed to work in pairs for this assignment, and you are strongly encouraged to do so, though it is not required.

– Every lab requires a README file. The required name of this file for this lab is lab4Readme [Case matters!]. Do not change the name, or your lab submission will be penalized 20%. The file should have EXACTLY this name, with no extensions, such as .txt, etc. The Linux OS can determine that the file is a text file without any extension. Just create a text file with the text editor of your choice on stdlinux, and save it with the name specified. This file should include the following:

- Student name(s) -to avoid the 10% deduction (as explained in the course syllabus). If working with a partner, be sure to include BOTH NAMES;
- Effort distribution for each contributor (assumed to be 100% if you are not working with a partner);
- Total amount of time (hours) to complete the entire lab;
- Short description of any concerns, interesting problems or discoveries encountered, or comments in general about the contents of the lab. Do not omit this, or simply write something such as “none,” or you will lose the credit for this part. Write at least a few sentences here related to concerns, interesting problems, and so on, as described above.

– You should always aim to hand an assignment in on time. If you are late (even by a minute), you will receive 75% of your earned points for the designated grade as long as the assignment is submitted by 11:30:00 pm the following day, based on the due date shown at the top of this document. If you are more than 24 hours late, you will receive a zero for the assignment and your assignment will not be graded at all.

– Any lab submitted that does not assemble (using the commands indicated below) and run WILL RECEIVE AN AUTOMATIC GRADE OF ZERO for the relevant part of the lab. No exceptions will be made for this rule; to achieve even a single point on a lab your code must minimally compile and/or assemble and execute without crashing.

To assemble your code, you must use the following command (these are the commands the graders will use also):

At the Linux command line prompt:

% yas lab4.ys

– You are welcome to do more than what is required by the assignment as long as it is clear what you are doing and it does not interfere with the mandatory requirements.

– You are responsible for making sure that your lab submits correctly.

– Be sure to include comments in the code documenting what the code does, and pay attention to formatting to make sure the code is as clear as possible, and easy for a reader of your program to understand.

GRADING CRITERIA (approximate percentages listed)

- (20%) The code and algorithm are well commented.
  - A comment should be included in the main program including the programmer name(s) as well as explaining the nature of the problem and an overall method of the solution (what you are doing, not how).
  - A short comment should be included for each logical or syntactic block of statements, including each function.
- (20%) The program should be appropriate to the assignment, well-structured and easy to understand, without complicated and confusing flow of control.
- (60%) The results are correct, verifiable, and well-formatted. The program correctly performs as assigned.

LAB DESCRIPTION

Y86 ASSEMBLY LANGUAGE PROGRAM TO FIND THE NUMBER OF ELEMENTS IN AN ARRAY, THE MIN ELEMENT, THE MAX ELEMENT, AND TO OUTPUT THE ELEMENTS IN REVERSE ORDER (96%). Mandatory filename: lab4.ys

PROBLEM:

Write an assembly language program with five procedures using Y86. The program should have an input array, defined at the top of the program, with an indeterminate number of values (but no more than 25), which appear in unsorted order. The program will need a Main procedure, a Count procedure, to determine the number of values in the input array, a Min procedure, to find the minimum value in the array, a Max procedure, to find the maximum value in the array, and a Reverse procedure, which will write the values in the input array to output in reverse order.

You are required to define the following at the top of your program:

(1) An input array label, Array; put this label at address 0x400.

(2) The actual array values (see below for sample input)

(3) A label at the end of the input array, Done, with no data value (that is, the label only)

(4) An output label, Output; put this label at address 0x500.

(5) A stack label, Stack, at a high enough address so that the stack will not grow into the output data, the input data, or the code; put this label at address 0xF00.

Example input, and memory for output and stack:

```
#Use appropriate .pos and .align directives here
Array:
        .long 12
        .long -2
        .long 16
        .long -10
        .long 13
        .long -27
        .long 10

Done:

#Use appropriate .pos and .align directives here

Output:

#Use appropriate .pos and .align directives here

Stack:
```

INPUT:

- There are no data errors that need to be checked as all the data will be assumed correct. You can assume that the values in the input array are signed integers, which may be negative, positive, or zero.
- Your program should be able to handle an unknown number of array values (up to 25) and still work. That is, you cannot assume that you know how many input values there will be. The grader will change the number of input values in the input array, and the values themselves, to test your program.

- You must use the Done label at the end of the input to detect the end of the input array.
- You should not change the names of the labels, but you can use any addresses you wish, as long as your program executes correctly, although the addresses given above will definitely work.

CONSTRAINTS:

- Your program must have five procedures, with the labels Main, Count, Min, Max, and Reverse.
- You need to write the Main procedure to call the other four procedures at appropriate points, and with the appropriate parameters on the stack to pass to the procedures, so that they do what is described below. You are allowed to do other work in Main to manipulate parameter values to pass to the procedures before you call them. Main should also write the values returned by all the procedures to output, except for the last procedure, Reverse, which will write its output directly without returning it to Main.
- The Count procedure should determine the number of values in the input array, and return this value to Main. Main will then write the value to output. This value should be in the first memory location in the output. Count should be passed exactly two parameters, namely, a pointer to Array, and the address of the Done label.
- The Min procedure should find the minimum value in the input array, and return this value to Main. Main will then write the value to output. This value should be in the second memory location in the output. Min should be passed exactly two parameters, namely, a pointer to Array, and the number of values in Array.
- The Max procedure should find the maximum value in the input array, and return this value to Main. Main will then write the value to output. This value should be in the third memory location in the output. Max should be passed exactly two parameters, namely, a pointer to Array, and the number of values in Array.
- The Reverse procedure should write the values in the input array to output in reverse order, i.e., from the last value in the input array to the first. Reverse should be passed exactly three parameters, namely, a pointer to Array, the number of values in Array, and the next address in the output area of memory where Reverse should begin writing the elements in Array in reverse order. *You cannot access the Done label or its address from procedure Reverse*. You cannot print the values in reverse by pushing all of them onto the stack and then popping them off one by one and writing them to output.
- To pass values to procedures, you must use the stack. You cannot pass parameters in registers. The parameters needed by a procedure can be pushed onto the stack in any order (for this lab; we will see later that there is a convention for the order). Remember that values which are returned by procedures should be returned in register *eax*.

You should also follow the IA32 conventions for caller save and callee save registers; that is, if a callee (a called function) needs to use a callee save register, it

should push it onto the stack before moving anything to it, and then pop it back off the stack before returning. Likewise, if a caller needs values in caller save registers after return, it should push them onto the stack before the call, and pop them back off the stack after return from the call.

OUTPUT:

- You need an output area of memory in your program which is required to be at the top of your program with the descriptive label given above associated with it.
- The output is the number of values in the input Array, the minimum value in the input array, and the maximum value in the input array, followed by all the values in the input array, written in reverse order. For the data given in the problem section above, the output, in hexadecimal, should be:

```
0x00000007      #Number of values in Array
0xfffffffe5     #Minimum value
0x00000010      #Maximum value
0x0000000a      #Remaining values are those in input array, in reverse order
0xfffffffe5
0x0000000d
0xfffffff6
0x00000010
0xfffffffe
0x0000000c
```

LAB SUBMISSION

You should submit all your lab assignments electronically using the submit command. The format of the submit command for this lab is as follows:

9:10 section: % submit c2421ab lab4 lab4Readme lab4.ys

10:20 section: % submit c2421ad lab4 lab4Readme lab4.ys

1:50 section: % submit c2421aa lab4 lab4Readme lab4.ys

NOTE:

- All of the files in a lab MUST be submitted using one command. If you use two submit commands, the files submitted by the second submit will erase the files from the first submission. Therefore, if you make changes to your files and decide to re-submit the lab assignment before the deadline, be sure to submit all of the required files with a single submit command. Do not press the enter key in the middle of your submit command line; let it wrap to the following line if necessary.

Your programs MUST be submitted in source code form. Make sure that you submit a .ys file. Do NOT submit the object file (.yo file).

- It is YOUR responsibility to make sure your code can assemble and run on the CSE department server stdlinux.cse.ohio-state.edu using yas and yis as specified earlier in this lab description.