Prof. Jingke Li (FAB 120-06, li@cs.pdx.edu), Classes: MW 16:40-18:30, URBAN 204, Office Hours: MW 14:00-15:00.

Project 4: SPARC Assembly Programming
(Due Wednesday 2/29/12)

This project is a programming exercise in the SPARC assembly language. It is to provide a background for the last compiler project, the code generator. Note that this project is **due in one week**.

General Description

Write a SPARC(V8) assembly function *scan*(*a*,*n*). It should take as input an one-dimensional integer array a of size n, and return a new integer array of size $n+1$, such that the ith element of the new array (for $i <= n$) equals the sum of the first i elements of a, and the last element represents the maximal element of a. For instance, if the input array consists of four elements $[1, 2, 3, 4]$, then the output array should have five elements $[1, 3, 6, 10, 4]$.

The function should be made suitable for calling from C with the following interface

```
int *scan(int *a, int n)
```

where `a` is a pointer to the input array and `n` is the size of the array. The scan funcion should first check to make sure that `n>0`; if not, it should return a null immediately. For the normal case, the function should allocate a fresh array of size $n+1$ on the heap to hold the result, and return a pointer to this array after filling it in.

For example, if your function is linked with the following driver program (which is available in `driver.c`):

```
extern int *scan(int *, int);

int main(void)
{
    int a[] = {1, 2, 3, 4, 5, 6, 7, -8};
    int i, *c;

    if ((c = scan(a, 8))) {
      printf("c = [");
      for (i=0; i<9; i++)
        printf(" %4d", c[i]);
      printf("]\n");
    } else {
      printf("Error in scan\n");
    }
}
```

it should produce the following output:

```
c = [    1    3    6   10   15   21   28   20    7]
```

Place your `scan` function in a file `scan.s`. It's starting lines should look like:

```
        .section ".text"
        .align 4
        .global scan
scan:
```

To test your program, you need to log on to a SPARC machine (*e.g.* `unix.cs.pdx.edu`), and compile `scan.s` and the driver program together:

```
gcc driver.c scan.s -o scan
```

and run `./scan`.

Implementation Requirements

- Include a comment block in `scan.s` listing
  - the algorithm your function implements (using C-like pseudo code)
  - the purpose of all registers that are used in the function
- Provide a comment for **every line of the assembly code** indicating what it does. This comment *must* be expressed in terms of the (pseudo) C code. For example, for the following lines in C code

```
x = 9;
y = x - 7;
z = f(x, y);
```

  the assembly code might contain the following commented lines:

```
mov     9, %o0          ! x = 9, first operand
call    f               ! call f, result in %o0
sub     %o0, 7, %o1     ! y = x - 7, second operand, **delay slot**
mov     %o0, %l1        ! store result in z
```

- Clearly mark *every* delay-slot instruction in its comment.

Your program will be graded on correctness, clarity in comments, and concision.

Hints

You may want to write the scan function in C first, and use `gcc` with switch `-S` to generate the corresponding assembly code. This can be very useful for seeing what instructions the compiler chooses, exactly what the function calling conventions are, etc. You are allowed to use this compiler-generated code as the base for your version. However, you still need to understand every line of this code in order to add comments. Furthermore, you may need to improve this code by hand to avoid being penalized for obvious stupidities in the compiler-generated assembly code.

Program Submission

Submit your fully commented `scan.s` file through the "Dropbox" on the D2L class webpage.