

Assignment 1, CSci 330, Fall 2006

Due: Sep 1, 2:10pm. Value: 40 pts.

The `strstr` C function searches for a substring within a bigger string. Suppose, for example, that a program calls `strstr("nefarious", "far")`; the function should return the the address of of 'f' within "nefarious", since the substring *far* begins with this letter. Should a program call `strstr("nefarious", "fas")`; the function should return `NULL`, since *fas* is not a substring within the source string. Here is a C implementation.

```
char* strstr(char *haystack, char *needle) {
    int i;
    int j;

    for(i = 0; haystack[i]; i++) {
        for(j = 0; needle[j] && needle[j] == haystack[i + j]; j++);
        if(!needle[j]) return haystack + i;
    }
    return NULL;
}
```

Your job is to write `strstr` in MIPS assembly language following loosely the algorithm reflected in the C implementation above. As a starting point, download from the course Web page the file `strstr.s`. It contains a stub implementation of `strstr`, which you are to complete, and a lot of code that will test `strstr` when the program is executed. You can ignore this extra code, although you are also welcome to modify it to add additional tests.

To assemble and execute your MIPS code, use *Mars*, downloadable from

<http://courses.missouristate.edu/KenVollmar/MARS/>

It is already installed on the Linux computers: Execute `mars` from the command line.

Your grade will be based on three components.

Correctness (40%) The subroutine works on all possible inputs. Note that it is possible that the test cases do not catch all possibilities, and that the above C implementation may have bugs (although I do not know of any). I will be very happy to hear about such.

Efficiency (30%) The program completes its job efficiently within the framework of the algorithm specified. In particular, I will count the number of instructions in the inner loop and the number of non-inner-loop instructions in the outer loop. The number for the inner loop will receive a larger weight.

Documentation (30%) Document your code thoroughly so that a reader (particularly I) can easily understand what you are doing. Note that more documentation does not necessarily help; and in no case should your documentation simply restate what an instruction obviously already says.

Submit your solution by e-mail to burch@grendel.hendrix.edu.