Universität Rostock Traditio et Innovatio

Institut für Angewandte Mikroelektronik und Datentechnik
Lehrstuhl für Eingebettete Systeme
Prof. Dr.-Ing. habil. Ch. Haubelt
Praktikum: Benjamin Beichler

Computer Organization

Assembler Practice Tasks 2

1 Usage of Subroutines 1 – strlen_register

Create a file `strlen_register.txt` with the code below. Use the code of `strlen.txt` from last session to create a subroutine at label `strlen`, which uses `a0` as input register for a string address and put the length of the string in `a0` into `a0`. Use the correct call convention and save all registers according to the lecture. Simulate the Program with RARS. Then extend the main routine to print length of string `B`.

```
.data
A:  .asciz   "ABCDEF"
B:  .asciz   "The quick brown fox jumps over the lazy dog"
LF: .asciz   "\n"

.text

main:
addi     sp, sp, -4
sw       ra, 0(sp)
la       s0, LF
la       a0, A        # Load A
jal      ra, strlen
li       a7, 1        # ecall 1 = print_int
ecall
mv       a0, s0       # line feed string into input register
li       a7, 4        # ecall 4 = print_string
ecall
lw       ra, 0(sp)
addi     sp, sp, 4
li       a7,10        # terminate ecall
ecall
strlen:
...
```

2 Usage of Subroutines 2 – strlen_stack

Use the Code of `strlen_register.txt` and create a file `strlen_stack.txt`. Modify the code to use the stack as source for input strings instead of the `a0` register. Simulate your code with RARS and check the results.

3 Recursion – Fibonacci

Calculate the corresponding Fibonacci number ($f(n) = f(n-1) + f(n-2)$ with $n_1 = 1; n_2 = 1; n_3 = 2$) of the global variable `N` with a recursive subroutine. Use `a0` as input value and `a0` for the return value. Create a main routine, that calculates $fibonacci(10)$, and print the result to the console. Store the resulting program in file `Fibonacci_recursive.txt` and simulate it with RARS.

4 Additional Task – quicksort

Edit the file `quicksort.txt`, which should implement the quicksort algorithm for an array of 32bit integers. The template defines the main program, with a routine to dynamically fill the array with random numbers. As a starting point for quicksort you could use the c-code below. Be aware of the byte addressed memory but word aligned memory access. Simulate your file with the given main function in RARS and check the results.

```
void quick_sort (int *a, int n) {
    if (n < 2)
        return;
    int p = a[n / 2];       // get pivot at half of range
    int *l = a;
    int *r = a + n - 1;
    while (l <= r) {
        if (*l < p) {     //search element bigger than pivot
            l++;
            continue;
        }
        if (*r > p) {     //search element smaller than pivot
            r--;
            continue;
            /* we need to check the condition (l <= r)
             * every time we change the value of l or r
             */
        }
        int t = *l;       //swap
        *l = *r;
        *r = t;
        ++l;
    }
    quick_sort(a, r - a);
    quick_sort(l, a + n - 1);
}
```