CSI 333 – Programming at the Hardware/Software Interface

A Complete MAL Program – Example 5

Note: This handout is referred to on page 10-9 of the lecture slides.

The following is an example of a MAL program containing two functions, namely main and sum. The function sum has one parameter and also a return value. Both C++ and MAL versions of these functions are given below.

---

**C++ Versions**

```c++
main() {
    int r = 10;
    int y = sum(r) + 9;
    cout << y << endl;
}

int sum (int x) {
    if (x <= 0)
        return 0;
    else
        return (x*(x+1)/2);
}
```

---

**MAL Versions**

```
#Function: main
#The variables r and y are stored in $5 and $6 respectively.
#The function sum expects one argument in $5 and returns a value in $7.

.text
.globl __start
__start:
    li $5, 10  #r is assigned 10.
    jal sum

    addi $6, $7, 9  #Compute y in $6.
    li $v0, 1  # code to print an integer
    move $a0, $6  # print the value of y
    syscall
    li $v0, 4  #print_string
    la $a0, nlst # newline
    syscall  # print_string done

    li $v0, 10
    syscall  # au revoir done

.data
nlst: .asciiz "\n"
```
# Function: sum
# The value of the parameter (x) is in $5. The return value will be in $7.

.text
sum:    bgtz $5, compute  # if $5 contains a number > 0 go to
        # code at compute, which will return with the
        # return value in $7.
    li $7, 0             # else, put the return value in $7
    jr $31               # and do the return.

# Need to compute the return value.
# Save and restore $6 and $8 so that they can be used as holders of
# temporary values during the computation.
compute: # parameter x is in $5
    addi $sp, $sp, -8    # Push 2 stack words
    sw $6, 4($sp)        # Store $6 in stack.
    sw $8, 0($sp)        # Store $8 in stack.
    addi $6, $5, 1       # $6 has the value x+1.
    mul $6, $6, $5       # $6 has the value x(x+1).
    li $8, 2             #
    div $7, $6, $8       # $7 has the value x(x+1)/2
                        # (i.e., the correct return value).
    lw $8, 0($sp)        # Restore $8.
    lw $6, 4($sp)        # Restore $6.
    addi $sp, $sp, 8     # Pop stack (twice).
    jr $31               # Return to main.