HPY201 Project

C → CLANG → MIPS Assembly

Vatsolakis Charalampos, 2018

Recursion in C

```
int factorial(int x){
    if(x==1)
        return 1;
    else
        return x*factorial(x-1);
}
```

C → intermediate C

```
int factorial(int x){
    if(x==1)
        return 1;
    else
        return x*factorial(x-1);
}
```

```
int factorial (int x){
    int ret;
    if (x==1) ret = 1;
    else {
        int f = factorial(x-1);
        int i, sum = 0;
        for (i=0;i<x; i++)
            sum = sum + f;
        ret = sum;
    }
    return ret;
}
```

intermediate C → CLANG

```
int factorial (int x){
    int ret;
    if (x==1) ret = 1;
    else {
        int f = factorial(x-1);
        int i, sum = 0;
        for (i=0;i<x; i++)
            sum = sum + f;
        ret = sum;
    }
    return ret;
}
```

```
void factorial(void){
    int temp[2];
    temp[0] = S0;
    S0 = A0;
    T0 = 1;
    if(A0 != T0) goto else_label;
    V0=1;
    goto end_if_label;
    else_label:
    A0 = S0 – 1;
    temp[1] = RA;
    factorial();
    RA = temp[1];
```

intermediate C → CLANG

```
int factorial (int x){
    int ret;
    if (x==1) ret = 1;
    else {
        int f = factorial(x-1);
        int i, sum = 0;
        for (i=0;i<x; i++)
            sum = sum + f;
        ret = sum;
    }
    return ret;
}
```

```
T0 = 0; // sum
T1 = 0; //i
repeat_for:
T2 = T1<S0;
if (T2 == 0) goto end_for;
T0 = T0 + V0;
T1 = T1 + 1;
goto repeat_for;
end_for:
V0 = T0;
end_if_label:
S0 = temp[0];
//FREE STACK
return;
}
```

CLANG → Assembly

```
void factorial(void){
    int temp[2];
    temp[0] = S0;
    S0 = A0;
    T0 = 1;
    if(A0 != T0) goto else_label;
    V0=1;
    goto end_if_label;
    else_label:
    A0 = S0 – 1;
    temp[1] = RA;
    factorial();
    RA = temp[1];
```

```
factorial:
addi $sp, $sp, -8
sw $s0, 0($sp)
add $s0, $a0, $zero
addi $t0, $zero, 1
bne $a0, $t0, else_label
addi $v0, $zero, 1
j end_if_label
else_label:
addi $a0, $s0, -1
sw $ra, 4($sp)
jal factorial
lw $ra, 4($sp)
```

CLANG → Assembly

```
    T0 = 0; // sum
    T1 = 0; //i
    repeat_for:
    T2 = T1<S0;
    if (T2 == 0) goto end_for;
    T0 = T0 + V0;
    T1 = T1 + 1;
    goto repeat_for;
    end_for:
    V0 = T0;
    end_if_label:
    S0 = temp[0];
    //FREE STACK
    return;
}
```

```
add $t0, $zero, $zero
add $t1, $zero, $zero
repeat_for:
slt $t2, $t1, $s0
beq $t2, $zero, end_for
add $t0, $t0, $v0
addi $t1, $t1, 1
j repeat_for
end_for:
add $v0, $t0, $zero
end_if_label:
lw $s0, 0($sp)
addi $sp, $sp, 8
jr $ra
```

C → Assembly

```
int factorial(int x){
    if(x==1)
        return 1;
    else
        return x*factorial(x-1);
}
```

```
factorial:
addi $sp, $sp, -8
sw $s0, 0($sp)
add $s0, $a0, $zero
addi $t0, $zero, 1
bne $a0, $t0, else_label
addi $v0, $zero, 1
j end_if_label
else_label:
addi $a0, $s0, -1
sw $ra, 4($sp)
jal factorial
lw $ra, 4($sp)
add $t0, $zero, $zero
add $t1, $zero, $zero
repeat_for:
slt $t2, $t1, $s0
beq $t2, $zero, end_for
add $t0, $t0, $v0
addi $t1, $t1, 1
j repeat_for
end_for:
add $v0, $t0, $zero
end_if_label:
lw $s0, 0($sp)
addi $sp, $sp, 8
jr $ra
```