ACS. Homework 5

Александр Васюков | БПИ235 (239)

Задание

`Разработать программу, определяющую максимальное значение аргумента, при котором результат вычисления факториала размещается в 32-х разрядном машинном слове. Вычисление факториала организовать как подпрограмму с циклом, которая возвращает найденный аргумент в регистре a0. Вывод результатов должна осуществлять главная функция. Дополнительно реализовать решение предыдущей задачи с использованием рекурсивной подпрограммы вычисления максимального значения аргумента, при котором результат вычисления факториала размещается в 32-х разрядном машинном слове.`

О программе find_max_factorial.asm

В секции `.data` находятся текстовые сообщения для вывода результата.

В `main` вызывается подпрограмма `fact`, которая вычисляет факториал, и выводится текст с результатом.

Подпрограмма `fact` на стеке выделяет 3 "ячейки" и сохраняет в них адрес возврата `ra`, аргумент факториала `s0`, факториал `s1`. Устанавливаются начальные значения, равные `1`, для аргумента и факториала.

В цикле `loop` увеличивается аргумент и сохраняются `ra`, `s0`, `s1`. Дальше аргумент и факториал перемещаются в регистры `a2` и `a3` соответственно для передачи их в качестве параметров в подпрограмму `check` для проверки, что факториал помещается в 32-разрядном машинном слове.

После проверки восстанавливаются значения `ra`, `s0`, `s1`.
Если `check` вернула `0` (переполнения не было), то умножаем `s0` на `s1`, результат помещается в `s0` и программа "прыгает" на цикл ещё раз.
Если же `check` вернула `1` (переполнение произошло), программа выходит из цикла:

1. Значение аргумента уменьшается на единицу, т.к. при нынешнем значении факториал не помещается в 32-разрядное машинное слово.
2. Восстанавливаются `ra`, `s0`, `s1`.

Подпрограмма `check` проверяет, помещается ли факториал в 32-разрядное машинное слово. Получая в качестве параметра значение аргумента `a2` и факториал `a3`, перемножает их и делит на `a2`. Если результат не равен `a3`, произошло переполнение при умножении, значит, факториал не удовлетворяет условию задачи и возвращается `0`. Если результат равен `a3`, умножение было корректным и возвращается `1`.

find_max_factorial.asm

GitHub: https://github.com/vasyukov1/HSE-FCS-SE-2-year/blob/main/ACS/Homework/5/find_max_factorial.asm

```
.data
res_p1: .asciz "LOOP SOLUTION\nThe largest argument whose factorial fits in 32-bit machine word is "
res_p2: .asciz ".\nFactorial: "

.text
main:
        # The first part of message for answer
        li a7 4
        la a0 res_p1
        ecall

        # Calculation
        jal fact

        # Result is already in a0
        # Output the argument of max factorial
        li a7 1
        ecall

        # The second part of message for answer
        li a7 4
        la a0 res_p2
        ecall
```

```
        # Output the value of max factorial
        li a7 1
        mv a0 a1
        ecall

        # Stop
        li a7 10
        ecall

fact:
        # Return: a0 - n (agrument of factorial), a1 - n! (value of factorial)
        # Reserve 3 numbers on stack for ra, argument and factorial
        addi sp sp -12
        sw ra 8(sp)
        sw s0 4(sp)
        sw s1 (sp)

        li s0 1         # Variable for the argument of factorial
        li s1 1         # Variable for the value of factorial

loop:
        addi s0 s0 1    # Increase argument
        sw ra 8(sp)
        sw s0 4(sp)
        sw s1 (sp)

        mv a2 s0
        mv a3 s1

        jal check

        lw ra 8(sp)
        lw s0 4(sp)
        lw s1 (sp)

        # If there isn't overflow, go next
        bnez a0 end_loop
        mul s1 s0 s1
        j loop

end_loop:
        # Return value of argument to correct after overflow
        addi a0 s0 -1
        mv a1 s1

        # Restoring
        lw ra 8(sp)
        lw s0 4(sp)
        lw s1 (sp)
        addi sp sp 12

        ret

check:  # Boolean function for checking overflow: 1 if overflow, else 0
        mul t0 a2 a3
        div t1 t0 a2

        bne t1 a3 overflow
        li a0 0
        ret
overflow:
        li a0 1
        ret
```

Результат работы

```
LOOP SOLUTION
The largest argument whose factorial fits in 32-bit machine word is 12.
Factorial: 479001600
-- program is finished running (0) --
```

О программе find_max_factorial.asm

Относительно предыдущей задаче изменений в секции `.data` и `main` нет, кроме отредактированного вывода результата (`RECURSIVE SOLUTION` вместо `LOOP SOLUTION`).

Подпрограмма `fact` выделяет 3 "ячейки" и сохраняет в них адрес возврата `ra`, аргумент факториала `s0`, факториал `s1`. Устанавливаются начальные значения, равные `1`, для аргумента `a2` и факториала `a3`. Именно в эти регистры, чтобы передать их как параметры в подпрограмму `recursive`. После завершения рекурсии значение аргумента корректируется, восстанвливаются значения из стека.

Подпрограмма `recursive`:

1. Увеличивается значение аргумента.
2. Считывается факториал из параметра `a3`.
3. Сохраняются адрес возврата, аргумент и факториал на стеке.
4. Вызывается подпрограмма проверки `check`.
5. Восстанавливаются адрес возврата, аргумент и факториал.
6. Если `check` вернула не `0`, значения аргумента и факториала перемещаются в регистры `a0` и `a1` и подпрограмма завершается.
7. Сохраняются адрес возврата, аргумент и факториал на стеке.
8. Вызывается подпрограмма проверки `recursive`.
9. Восстанавливаются адрес возврата, аргумент и факториал.

Подпрограмма `check` аналогична предыдущей задаче.

find_max_factorial_recursive.asm

GitHub: https://github.com/vasyukov1/HSE-FCS-SE-2-year/blob/main/ACS/Homework/5/find_max_factorial_recursive.asm

```
.data
res_p1: .asciz "RECURSIVE SOLUTION\nThe largest argument whose factorial fits in 32-bit machine word is "
res_p2: .asciz ".\nFactorial: "

.text
main:
        # The first part of message for answer
        li a7 4
        la a0 res_p1
        ecall

        # Calculation
        jal fact

        # Result is already in a0
        # Output the argument of max factorial
        li a7 1
        ecall

        # The second part of message for answer
        li a7 4
        la a0 res_p2
        ecall

        # Output the value of max factorial
        li a7 1
        mv a0 a1
        ecall

        # Stop
        li a7 10
        ecall

fact:
        # Return: a0 - n (agrument of factorial), a1 - n! (value of factorial)
        # Reserve 3 numbers on stack for ra, argument and factorial
        addi sp sp -12
        sw ra 8(sp)
        sw s0 4(sp)
        sw s1 (sp)

        li a2 1         # Setting the initial value of the argument
        li a3 1         # Setting the initial value of the factorial
```

```
        call recursive
        addi a0 a0 -1   # Adjusting the value of the argument

        # Restoring
        lw ra 8(sp)
        lw s0 4(sp)
        lw s1 (sp)
        addi sp sp 12

        ret

recursive:
        # Increasing the value of the argument
        addi s0 a2 1
        mv s1 a3

        # Saving
        addi sp sp -12
        sw ra 8(sp)
        sw s0 4(sp)
        sw s1 (sp)

        call check

        # Restoring
        lw ra 8(sp)
        lw s0 4(sp)
        lw s1 (sp)
        addi sp sp 12

        # If there is the overflow, go out
        bnez a0 end

        # Saving
        addi sp sp -12
        sw ra 8(sp)
        sw s0 4(sp)
        sw s1 (sp)

        # Increasing the factorial
        mv a2 s0
        mul a3 s1 s0

        call recursive

        # Restoring
        lw ra 8(sp)
        lw s0 4(sp)
        lw s1 (sp)
        addi sp sp 12

        ret
end:
        mv a0 s0
        mv a1 s1
        ret

check:  # Boolean function for checking overflow: 1 if overflow, else 0
        mul s0 a2 a3
        div s1 s0 a2

        bne s1 a3 overflow
        li a0 0
        ret
overflow:
        li a0 1
        ret
```

Результат работы

```
RECURSIVE SOLUTION
The largest argument whose factorial fits in 32-bit machine word is 12.
Factorial: 479001600
-- program is finished running (0) --
```