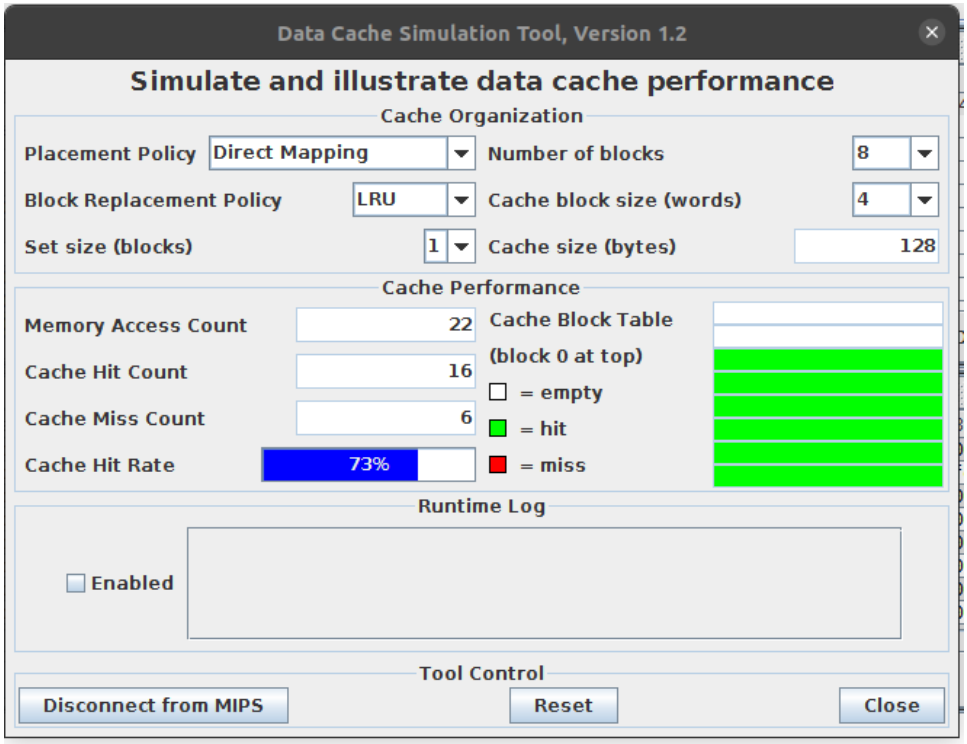
Assignment 4

- Mã nguồn

assign4.asm

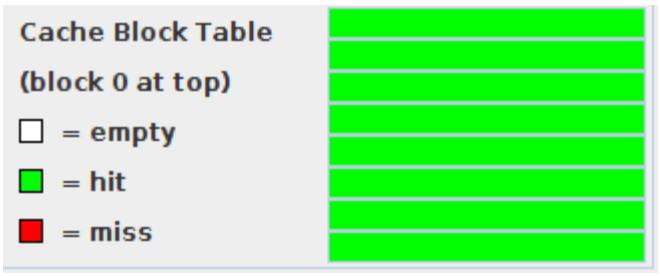
```
#Laboratory Exercise 7 Home Assignment 4

.data
Message: .asciiz "Ket qua tinh giai thua la: "
.text
main:   jal WARP
print:  add $a1, $v0, $zero # $a0 = result from N!
        li $v0, 56
        la $a0, Message
        syscall
quit:   li $v0, 10 #terminate
        syscall
endmain:
#---------------------------------------------------------------
#Procedure WARP: assign value and call FACT
#---------------------------------------------------------------
WARP:   sw $fp,-4($sp) #save frame pointer (1)
        addi $fp,$sp,0 #new frame pointer point to the top (2)
        addi $sp,$sp,-8 #adjust stack pointer (3)
        sw $ra,0($sp) #save return address (4)
        li $a0,6 #load test input N
        jal FACT #call fact procedure
        nop
        lw $ra,0($sp) #restore return address (5)
        addi $sp,$fp,0 #return stack pointer (6)
        lw $fp,-4($sp) #return frame pointer (7)
        jr $ra
wrap_end:
#---------------------------------------------------------------
#Procedure FACT: compute N!
#param[in] $a0 integer N
#return $v0 the largest value
#---------------------------------------------------------------
FACT:   sw $fp,-4($sp) #save frame pointer

        addi $fp,$sp,0 #new frame pointer point to stack's top
        addi $sp,$sp,-12 #allocate space for $fp,$ra,$a0 in stack
        sw $ra,4($sp) #save return address
        sw $a0,0($sp) #save $a0 register
        slti $t0,$a0,2 #if input argument N < 2
        beq $t0,$zero,recursive#if it is false ((a0 = N) >=2)
        nop
        li $v0,1 #return the result N!=1
        j done
        nop
recursive:
        addi $a0,$a0,-1 #adjust input argument
        jal FACT #recursive call
        nop
        lw $v1,0($sp) #load a0
        mult $v1,$v0 #compute the result
        mflo $v0
done:   lw $ra,4($sp) #restore return address
        lw $a0,0($sp) #restore a0
        addi $sp,$fp,0 #restore stack pointer
        lw $fp,-4($sp) #restore frame pointer
        jr $ra #jump to calling
fact_end:

```

- Màn hình chạy

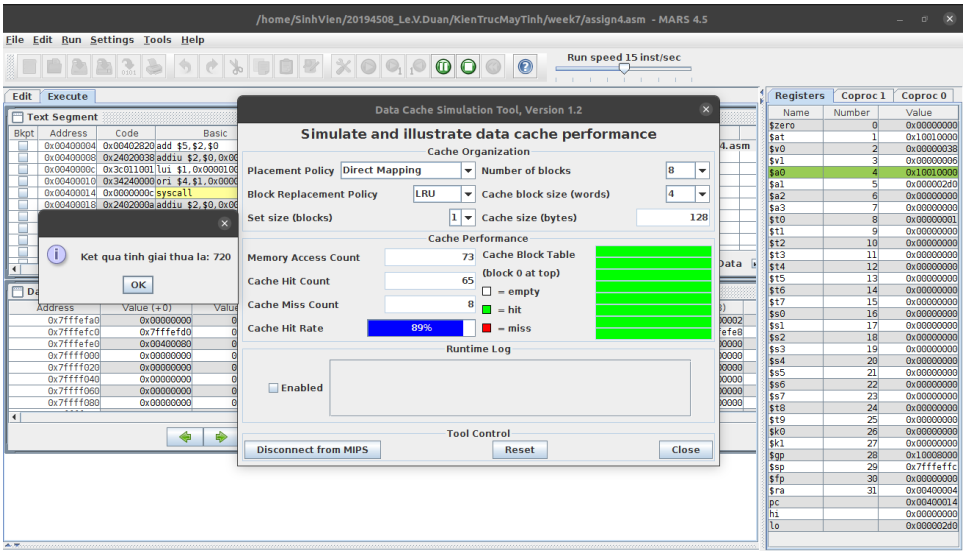


1. Cache Hit Count, Cache Miss count, ...

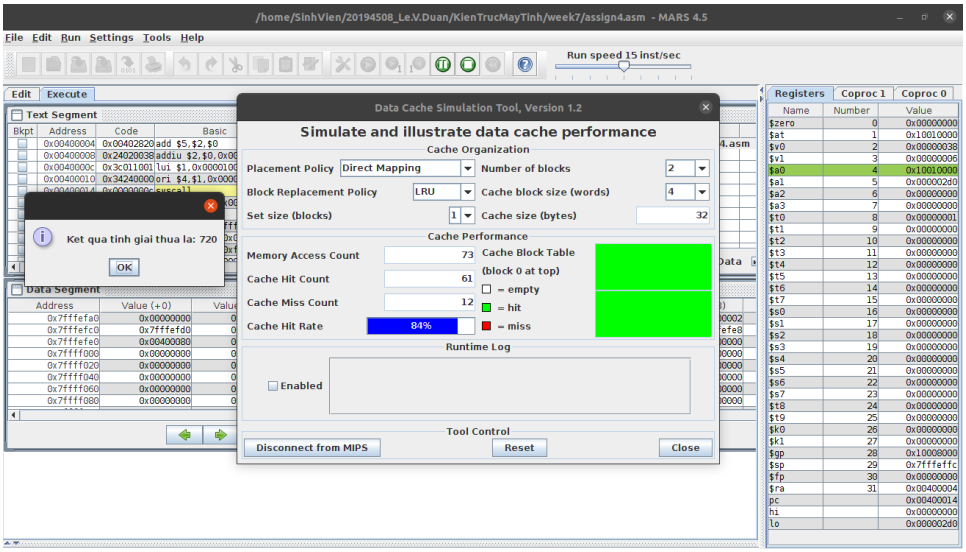
- Cache Hit count: số lần CPU yêu cầu truy cập vào Cache Memory thành công
- Cache Miss count: số lần CPU yêu cầu truy cập vào Cache Memory thất bại và phải truy cập vào bộ nhớ chính (Ram)
- Memory Access Count: số lần CPU yêu cầu truy cập vào Cache Memory (Tổng của Hit và Miss)
- Cache Hit Rate: tỷ lệ truy cập thành công vào Cache Memory (= Hit/Memory Access Count)
- Number of Blocks: số lượng block cho Cache Memory
- Cache block size (Words) : kích thước hay dung lượng của 1 block, như trong Mips mặc địch words là 4 bytes
- Cache size (bytes): kích thước hay dung lượng của Cache Memory (= Number of blocks * Cache block size)
- Biểu diễn và mô tả hoạt động của Cache được hiển thị tại Cache Block table:



- Khi Number of block tăng lên thì dẫn đến kích thước của Cache tăng lên -> Cache Hit count tăng lên và Cache Miss count giảm xuống.
- Ví dụ dưới đây thể hiển sự thay đổi khi chạy cùng 1 mã nguồn và thay đổi giá trị Number of block = 8



- Khi giảm Number of blocks xuống 2:



- Cache Hit Rate giảm từ 89% xuống 84%
- Cache Hit count: 65 -> 61
- Cache Miss count: 8 -> 12