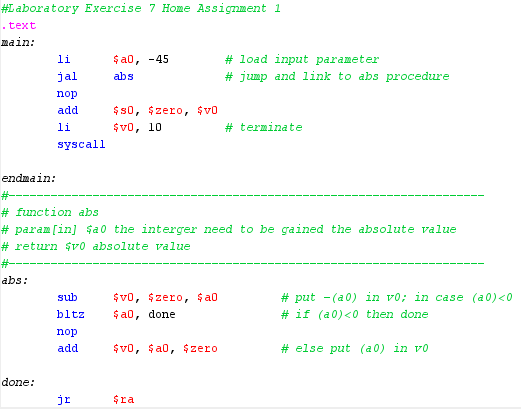
**BÁO CÁO THỰC HÀNH KIẾN TRÚC MÁY TÍNH (IT3280) TUẦN 7**

***Họ và tên: Phạm Đức Long***

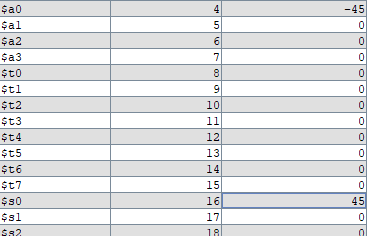
***MSSV: 20225737***

Assignment 1:

* Code:



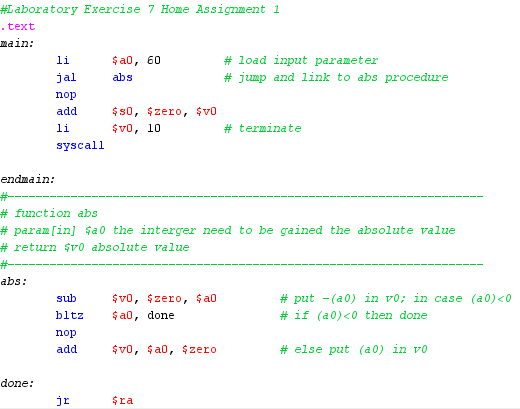
* Kết quả thu được:



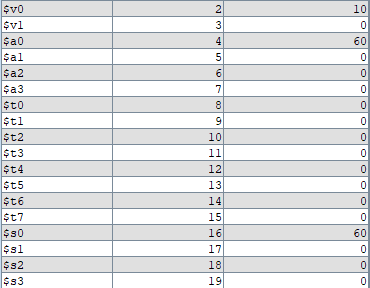
* Sự thay đổi của thanh ghi:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | $v0 | $s0 | $ra | $pc |
| Trạng thái ban đầu | 0x00000000 | 0x00000000 | 0x00000000 | 0x00400000 |
| Sau lệnh jal | 0x00000000 | 0x00000000 | 0x00400008 | 0x00400018 |
| Sau lệnh sub | 0x0000002d | 0x00000000 | 0x00400008 | 0x0040001c |
| Sau lệnh bltz | 0xffffffd3 | 0x00000000 | 0x00400008 | 0x00400028 |
| Sau lệnh jr | 0xffffffd3 | 0x00000000 | 0x00400008 | 0x00400008 |
| Sau lệnh add | 0xffffffd3 | 0xffffffd3 | 0x00400008 | 0x00400010 |
| Sau lệnh syscall | 0x0000000a | 0xffffffd3 | 0x00400008 | 0x00400018 |

* Với trường hợp tham số đầu vào dương:



* Kết quả thu được:



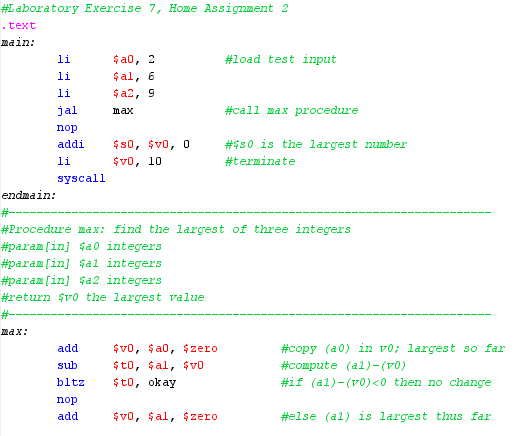
* Sự thay đổi của thanh ghi:

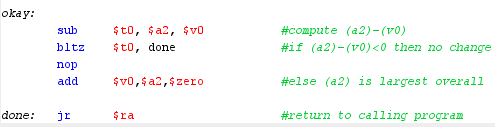
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | $v0 | $s0 | $ra | $pc |
| Trạng thái ban đầu | 0x00000000 | 0x00000000 | 0x00000000 | 0x00400000 |
| Sau lệnh jal | 0x00000000 | 0x00000000 | 0x00400008 | 0x00400018 |
| Sau lệnh sub | 0xffffffc4 | 0x00000000 | 0x00400008 | 0x0040001c |
| Sau lệnh bltz | 0xffffffc4 | 0x00000000 | 0x00400008 | 0x00400020 |
| Sau lệnh add | 0x0000003c | 0x00000000 | 0x00400008 | 0x00400028 |
| Sau lệnh jr | 0x0000003c | 0x00000000 | 0x00400008 | 0x00400008 |
| Sau lệnh add | 0x0000003c | 0x0000003c | 0x00400008 | 0x00400010 |
| Sau lệnh syscall | 0x0000000a | 0x0000003c | 0x00400008 | 0x00400018 |

* Giải thích:
* Chương trình truyền một tham số đầu vào và lưu vào thanh ghi $a0. Sau đó lệnh jal sẽ nhảy đến địa chỉ của nhãn cần nhảy đến (là nhãn abs) đồng thời lưu địa chỉ của câu lệnh ngay sau câu lệnh jal (là lệnh nop) vào thanh ghi $ra.
* Sau khi nhảy đến nhãn abs, chương trình thực hiện lưu giá trị -(a0) vào thanh ghi $v0, sau đó so sánh $a0 với 0. Nếu giá trị thanh ghi $a0 nhỏ hơn 0 thì $v0 không đổi và nhảy đến thẻ done, sau đó thực hiện lệnh jr để nhảy đến địa chỉ mà thanh ghi đó trỏ tới rồi cập nhật giá trị tuyệt đối vào $s0. Ngược lại nếu $a0 lớn hơn 0 thì cập nhật lại $v0 = $a0 và nhảy về vị trí cũ và cập nhật kết quả vào $s0.
* Nhận xét: Kết quả cuối cùng là giá trị tuyệt đối của tham số đầu vào được lưu vào thanh ghi $s0 và đã đúng như tính toán theo lý thuyết.

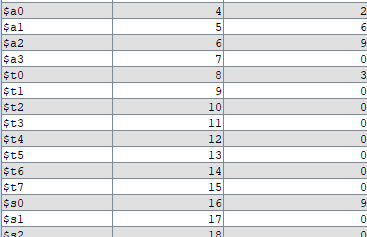
Assignment 2:

* Code:





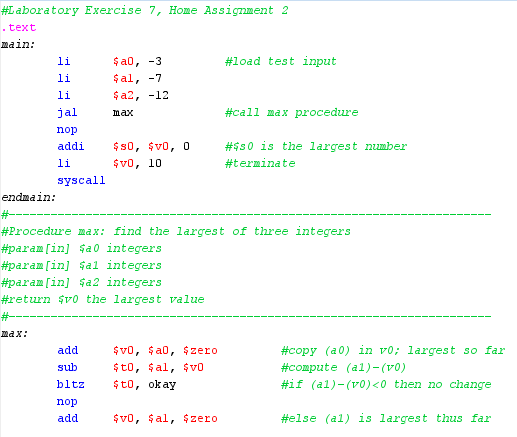
* Kết quả thu được:

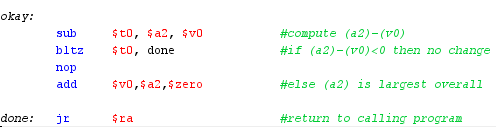


* Sự thay đổi giá trị thanh ghi:

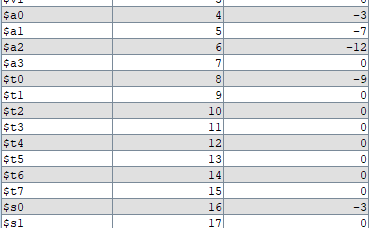
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | $v0 | $s0 | $ra | $pc |
| Trạng thái ban đầu | 0x00000000 | 0x00000000 | 0x00000000 | 0x00400000 |
| Sau khi khởi tạo giá trị | 0x00000000 | 0x00000000 | 0x00000000 | 0x0040000c |
| Sau lệnh jal | 0x00000000 | 0x00000000 | 0x00400010 | 0x00400020 |
| Sau lệnh add | 0x00000002 | - | - | 0x00400024 |
| Sau lệnh sub | - | - | - | 0x00400028 |
| Sau lệnh bltz | - | - | - | 0x0040002c |
| Sau lệnh add | 0x00000006 | - | - | 0x00400034 |
| Sau lệnh sub | - | - | - | 0x00400038 |
| Sau lệnh bltz | - | - | - | 0x0040003c |
| Sau lệnh add | 0x00000009 | - | - | 0x00400044 |
| Sau lệnh jr | - | - | - | 0x00400014 |
| Sau lệnh addi | - | 0x00000009 | - | 0x00400018 |
| Kết thúc chương trình | | | | |

* Giải thích:
* Chương trình truyền vào ba giá trị 2, 6, 9 vào 3 thanh ghi $a0, $a1, $a2
* Sau đó câu lệnh jal nhảy đến nhãn max và lưu địa chỉ của lệnh tiếp theo (nop) vào thanh ghi $ra
* Nhãn max truyền giá trị thanh ghi $a0 vào thanh ghi $v0, sau đó so sánh giá trị thanh ghi $a1 và $v0. Nếu $a1 lớn hơn thì cập nhật lại $v0 = $a1 rồi sau đó sẽ so sánh $v0 và $a2, nếu $a2 lớn hơn thì cập nhật lại $v0 = $a2. Còn nếu $a1 nhỏ hơn thì sẽ so sánh $a2 và $v0 để tìm ra giá trị lớn hơn, sau đó nhảy đến nhãn done và nhảy đến câu lệnh ngay sau câu lệnh jal, rồi gán trị trị lớn nhất đó vào $s0.
* Nhận xét: Kết quả cuối cùng là giá trị lớn nhất của ba tham số đầu vào, được lưu vào thanh ghi $s0 và đã đúng như tính toán theo lý thuyết.
* Với trường hợp khác:





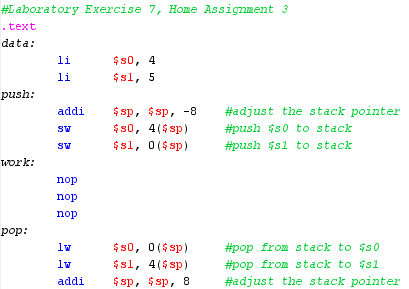
* Kết quả:



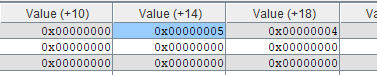
* Nhận xét: Giá trị lớn nhất là -3, được lưu vào thanh ghi $s0 🡪 Thỏa mãn.

Assignment 3:

* Code:



* Kết quả:
* Sau hai lệnh sw:



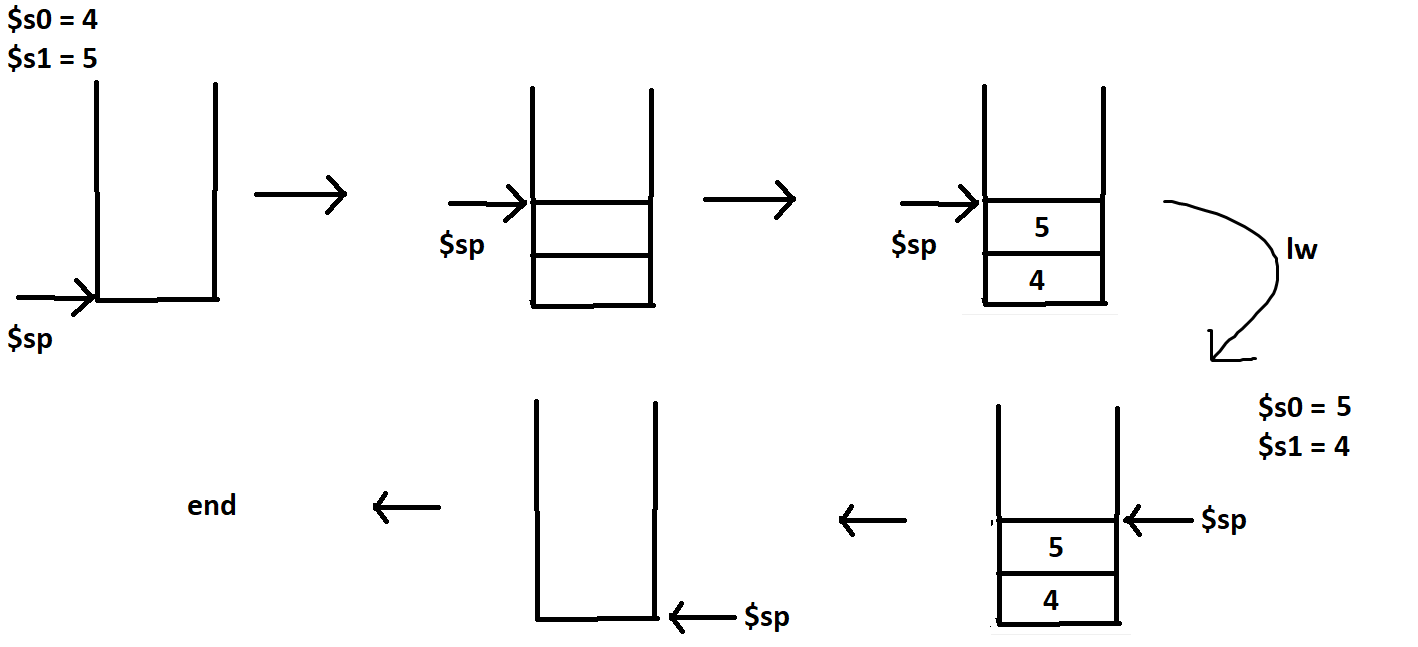
* Kết quả sau khi chạy hết chương trình:



* Sự thay đổi giá trị thanh ghi:

|  |  |  |  |
| --- | --- | --- | --- |
|  | $s0 | $s1 | $sp |
| Trạng thái ban đầu | 0x00000000 | 0x00000000 | 0x7fffeffc |
| Sau khi khởi tạo | 0x00000004 | 0x00000005 | - |
| Sau lệnh addi | - | - | 0x7fffeff4 |
| Sau hai lệnh sw | - | - | - |
| Sau hai lệnh lw | 0x00000005 | 0x00000004 | - |
| Sau lệnh addi | 0x00000005 | 0x00000004 | 0x7fffeffc |
| Kết thúc chương trình | | | |

* Giải thích:
* Chương trình dùng để hoán đổi 2 số bằng cách dùng stack: Sau khi khởi tạo giá trị hai thanh ghi $s0 và $s1 thì thực hiện hàm push chuyển vị trí con trỏ stack hiện tại về địa chỉ 2 ô trước đấy, đẩy giá trị của $s0 vào địa chỉ bên phải của stack hiện tại sau đó đẩy giá trị $s1 vào địa chỉ của stack hiện tại
* Hàm pop dùng để lấy giá trị từ stack truyền lại vào $s0 và $s1, truyền giá trị của địa chỉ stack hiện tại vào $s0 và giá trị của địa chỉ bên tay phải vào $s1, sau đó trả lại giá trị của ban đầu của stack
* Minh họa:



* Nhận xét: Chương trình đã đảo giá trị của $s0 và $s1 🡪 Thỏa mãn.

Assignment 4:

* Code với input:

**#Laboratory Exercise 7, Home Assignment 4**

**.data**

**Message: .asciiz "Ket qua tinh giai thua la: "**

**Message1: .asciiz "Nhap so can tinh giai thua:"**

**.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 $v0, 51 #inputDialogInt**

**la $a0, Message1**

**syscall**

**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**

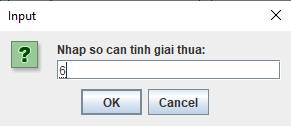
**addi $sp, $fp, 0 #restore stack pointer**

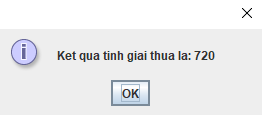
**lw $fp, -4($sp) #restore frame pointer**

**jr $ra #jump to calling**

**fact\_end:**

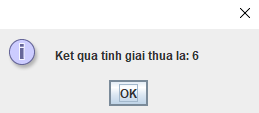
* Kết quả:
* $a0 = 6 (Trường hợp của Home Assignment 4):





* $a0 = 3 (Trường hợp của Assginment 4):





* Sự thay đổi giá trị thanh ghi với $a0 = 3:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | $ra | $sp | $fp | $pc |
| Trạng thái ban đầu | 0 | 2147479548 | 0 | 4194304 |
| Sau jal WARP | 4194308 | 2147479548 | 0 | 4194336 |
| Sau jal FACT (1) | 4194360 | 2147479540 | 2147479548 | 4194380 |
| Sau jal FACT (2) | 4194332 | 2147479528 | 2147479540 | 4194380 |
| Sau jal FACT (3) | 4194332 | 2147479516 | 2147479528 | 4194380 |
| Sau j done | 4194332 | 2147479504 | 2147479516 | 4194348 |
| Sau jr $ra (1) | 4194332 | 2147479516 | 2147479528 | 4194332 |
| Sau jr $ra (2) | 4194332 | 2147479528 | 2147479540 | 4194380 |
| Trước jr $ra (3) | 4194360 | 2147479540 | 2147479548 | 4194364 |
| Sau jr $ra (3) | 4194360 | 2147479540 | 2147479548 | 4194360 |
| In kết quả và kết thúc chương trình | | | | |

* Hình vẽ mô phỏng nguyên lý hoạt động của chương trình:

Assignment 5:

* Code:

**.data**

**a: .word 0**

**A: .space 32**

**Message: .asciiz "Nhap so $s"**

**MessageNum: .asciiz ": "**

**Message1: .asciiz "Gia tri lon nhat la: "**

**Message2: .asciiz " o thanh ghi $s"**

**Message3: .asciiz "Gia tri nho nhat la: "**

**Newline: .asciiz "\n"**

**.text**

**start:**

**la $a0, A # $a0 = address of A[0]**

**addi $t0, $a0, 0 # $50 = $a0**

**input:**

**beq $t1, 8, end\_input # if i = 8 end\_input**

**li $v0, 4**

**la $a0, Message**

**syscall**

**li $v0, 1**

**move $a0, $t1**

**syscall**

**li $v0, 4**

**la $a0, MessageNum**

**syscall**

**li $v0, 5**

**syscall # input number**

**move $t2, $v0**

**sw $t2, 0($t0) # $t0 = A[i]**

**addi $t0, $t0, 4 # address of A[i]**

**addi $t1, $t1, 1 # i++**

**j input**

**end\_input:**

**la $t0, A**

**addi $a0, $zero, 0**

**addi $t1, $zero, 0**

**addi $t2, $zero, 0 # reset register**

**load\_value:**

**lw $s0, 0($t0) # load $s0**

**lw $s1, 4($t0) # load $s1**

**lw $s2, 8($t0) # load $s2**

**lw $s3, 12($t0) # load $s3**

**lw $s4, 16($t0) # load $s4**

**lw $s5, 20($t0) # load $s5**

**lw $s6, 24($t0) # load $s6**

**lw $s7, 28($t0) # load $s7**

**main: jal WARP**

**print:**

**add $a1, $v0, $zero**

**li $v0, 4**

**la $a0, Message1**

**syscall**

**li $v0, 1**

**addi $a0, $a1, 0**

**syscall**

**li $v0, 4**

**la $a0, Message2**

**syscall**

**li $v0, 1**

**addi $a0, $t2, 0**

**syscall # print lagerst**

**li $v0, 4**

**la $a0, Newline**

**syscall**

**la $a0, Message3**

**syscall**

**li $v0, 1**

**addi $a0, $v1, 0**

**syscall**

**li $v0, 4**

**la $a0, Message2**

**syscall # print smallest**

**li $v0, 1**

**addi $a0, $t3, 0**

**syscall**

**quit:**

**li $v0, 10 # terminate**

**syscall**

**endmain:**

**WARP:**

**la $t0, a # address of A[-1]**

**addi $a0, $a0, -1 # j = -1**

**sw $fp, -4($sp) # save frame pointer**

**addi $fp, $sp, 0 # new frame pointer to top**

**addi $sp, $sp, -8 # next stack**

**sw $ra, 0($sp) # save return adress**

**jal stack**

**nop**

**lw $ra, 0($sp) # restore address from stack**

**addi $sp, $fp, 0 # return stack pointer**

**lw $fp, -4($sp) # return frame pointer**

**jr $ra**

**wrap\_end:**

**stack: sw $fp,-4($sp) # save frame pointer**

**addi $fp,$sp,0 # new frame pointer to top**

**addi $sp,$sp,-16 # create space for $ra, $a0, $A[i]( value of register s(j)**

**sw $ra,8($sp) # save return address**

**sw $a0,4($sp) # save number of register save value**

**lw $t1,0($t0) # $t1 = A[i] = value of s(j)**

**sw $t1,0($sp) # save s(j)**

**bne $a0,7,recursive #if j = 7 recursive**

**nop**

**lw $v0, 0($sp) # save max value**

**lw $v1, 0($sp) # save min value**

**lw $t2, 4($sp) # save number of register save max value**

**lw $t3, 4($sp) # save number of register save min value**

**j min\_max**

**nop**

**recursive:**

**addi $a0, $a0, 1 # j++**

**addi $t0, $t0, 4 # address of A[j]**

**jal stack**

**nop**

**j find\_max**

**nop**

**min\_max:**

**lw $ra, 8($sp) # save return address**

**lw $t1, 0($sp) # save temp value**

**lw $t4, 4($sp) # save temp number of register**

**addi $sp, $fp, 0 # restore stack pointer**

**lw $fp, -4($sp) # restore frame pointer**

**jr $ra**

**find\_max:**

**bgt $t1, $v0, max # if temp\_value > max**

**j find\_min**

**max:**

**addi $v0, $t1, 0 # max = temp\_value**

**addi $t2, $t4, 0 # number of register = temp number of register**

**nop**

**find\_min:**

**blt $t1, $v1, min # if temp\_vale < min**

**j min\_max**

**min:**

**addi $v1, $t1, 0 # min = temp value**

**addi $t3, $t4, 0 # number of register = temp number of register**

**j min\_max**

**nop**