Thực hành kiến trúc máy tính

Báo cáo thực hành

Bài 7. Lệnh gọi chương trình con, truyền tham số sử dụng ngăn xếp

Họ Tên	Lê Thành An
MSSV	20235631

ASSIGNMENT 1

ĐOẠN MÃ :

```
# Laboratory Exercise 7 Home Assignment 1
.text
   main:
   li  a0, -19
   # load input parameter
   jal abs
   # jump and link to abs procedure
   li  a7, 10
   # terminate
   ecall
end_main:
   # ---------------------------------------------------------------------
   # function abs
   # param[in]    a0     the interger need to be gained the absolute
value
   # return       s0     absolute value
   # ---------------------------------------------------------------------
abs:
   sub s0, zero, a0    # put -a0 in s0; in case a0 < 0
   blt a0, zero, done  # if a0<0 then done
   add s0, a0, zero    # else put a0 in s0
done:
   jr  ra
```

Kết quả:

STT	Vị trí	Thanh ghi	Giá trị thanh ghi	Ghi chú
1	main:	pc	0x00400000	Bắt đầu chương trình tại địa chỉ main.
2	li a0, -19	a0	0xFFFFFFED	Gán giá trị -19 (0xFFFFFFED) vào thanh ghi a0.
3	jal abs	ra	0x00400008	Nhảy đến abs, lưu địa chỉ trở về (0x00400008) vào ra.

4	abs:	pc	0x00400010	Địa chỉ bắt đầu của hàm abs.
5	sub s0, zero, a0	s0	0x00000013	Tính -a0 (0x13 = 19), lưu vào s0.
6	blt a0, zero, done	pc	0x00400018	So sánh a0 (0xFFFFFFED < 0), nhảy đến done.
7	done:	pc	0x0040001C	Nhảy đến nhãn done.
8	jr ra	pc	0x00400008	Nhảy về địa chỉ trong ra (0x00400008), trở về sau lời gọi jal abs.
9	li a7, 10	a7	0x0000000A	Gán giá trị 10 (syscall exit) vào a7.
10	ecall			Gọi hệ thống kết thúc chương trình.

ASSIGNMENT 2

ĐOẠN MÃ :

```
# Laboratory Exercise 7, Home Assignment 2
.text
main:
    li    a0, -2     # load test input
    li    a1, 12
    li    a2, 9
    jal   max       # call max procedure

    li    a7, 10    # terminate
    ecall
end_main:

# ---------------------------------------------------------------------------
# Procedure max: find the largest of three integers
# param[in]  a0  integers
# param[in]  a1  integers
# param[in]  a2  integers
# return     s0   the largest value
# ---------------------------------------------------------------------------
max:
    add     s0, a0, zero    # copy a0 in s0; largest so far
    sub     t0, a1, s0      # compute a1 - s0
    blt     t0, zero, okay  # if a1 - v0 < 0 then no change
    add     s0, a1, zero    # else a1 is largest thus far
okay:
    sub     t0, a2, s0      # compute a2 - v0
    blt    t0, zero,done  # if a2 - v0 <0 then no change
    add     s0, a2, zero    # else a2 is largest overall
done:
    jr      ra              # return to calling program
```

Kết quả:

STT	Vị trí	Thanh ghi	Giá trị thanh ghi	Ghi chú
1	main:	pc	0x00010000	Bắt đầu chương trình tại main

2	li a0, -2	a0	0xFFFFFFFE	Khởi tạo a0 = -2 (giá trị âm trong complement 2)
3	li a1, 12	a1	0x0000000C	Khởi tạo a1 = 12
4	li a2, 9	a2	0x00000009	Khởi tạo a2 = 9
5	jal max	ra	0x00010010	Nhảy đến max, lưu địa chỉ trở về (pc + 4 = 0x00010010) vào ra
6	max:	pc	0x00010020	Địa chỉ bắt đầu của hàm max (giả định)
7	add s0, a0, zero	s0	0xFFFFFFFE	Gán s0 = a0 = -2 (giá trị lớn nhất tạm thời)
8	sub t0, a1, s0	t0	0x0000000E	Tính t0 = a1 - s0 = 12 - (-2) = 14
9	blt t0, zero, okay	pc	0x00010028	Do t0 = 14 > 0, không nhảy, chạy lệnh tiếp theo
10	add s0, a1, zero	s0	0x0000000C	Cập nhật s0 = a1 = 12 (giá trị lớn nhất mới)
11	okay:	pc	0x0001002C	Nhãn okay (bỏ qua nếu không nhảy)
12	sub t0, a2, s0	t0	0xFFFFFFFD	Tính t0 = a2 - s0 = 9 - 12 = -3
13	blt t0, zero, done	pc	0x00010034	Do t0 = -3 < 0, nhảy đến done
14	done:	pc	0x00010034	Nhãn done (bỏ qua lệnh add s0, a2, zero)
15	jr ra	pc	0x00010010	Nhảy về địa chỉ trong ra (trở về main)
16	li a7, 10	a7	0x0000000A	Chuẩn bị gọi syscall để kết thúc chương trình
17	ecall			Kết thúc chương trình

ASSIGNMENT 3

ĐOẠN MÃ :

```
# Laboratory Exercise 7, Home Assignment 3
.text
khaibao :
   li s0, 9
   li s1, 6
push:
   addi sp, sp, -8 # adjust the stack pointer
   sw s0, 4(sp) # push s0 to stack
   sw s1, 0(sp) # push s1 to stack
work:
   nop
   nop
   nop
pop:
   lw s0, 4(sp) # pop from stack to s0
   lw s1, 0(sp) # pop from stack to s1
   addi sp, sp, 8 # adjust the stack pointer
```

Kết quả:

STT	Vị trí	Thanh ghi	Giá trị thanh ghi	Giá trị vùng nhớ stack	Ghi chú
0		sp	0x7fffeffc		Trạng thái ban đầu của thanh ghi sp
1	khaibao:	pc	0x00400000		Bắt đầu chương trình.
2	li s0, 9	s0	0x00000009		Gán s0 = 9.
3	li s1, 6	s1	0x00000006		Gán s1 = 6.
4	push:	pc	0x00400008		Chuẩn bị thao tác stack.
5	addi sp, sp, -8	sp	0x7fffeff4	[sp+4] và [sp]chưa xác định	Dịch con trỏ stack (sp) xuống 8 byte để dành chỗ cho s0 và s1.
6	sw s0, 4(sp)			[sp+4] = 0x00000009	Lưu s0 (9)vào địa chỉ sp + 4.
7	sw s1, 0(sp)			[sp] = 0x00000006	Lưu s1 (6) vào địa chỉ sp.
8	work:	pc	0x00400014		Các lệnh nop không làm thay đổi trạng thái.
9	pop:	pc	0x00400020		Bắt đầu khôi phục giá trị từ stack.
10	lw s0, 4(sp)	s0	0x00000006	[sp] = 0x00000009	Khôi phục giá trị từ [sp+4] vào s0
11	lw s1, 0(sp)	s1	0x00000009	[sp+4] = 0x00000006	Khôi phục giá trị từ [sp] vào s1
12	addi sp, sp, 8	sp	0x7fffeffc		Khôi phục con trỏ stack về trạng thái ban đầu.

ASSIGNMENT 4

ĐOẠN MÃ :

```
# Laboratory Exercise 7, Home Assignment 4
.data
message: .asciz "Ket qua tinh giai thua la: "
.text
main:
   jal WARP
print:
   add a1, s0, zero # a0 = result from N!
   li a7, 56
   la a0, message
   ecall
quit:
   li a7, 10   # terminate
   ecall
end_main:
# ---------------------------------------------------------------
# Procedure WARP: assign value and call FACT
# ---------------------------------------------------------------
WARP:
   addi sp, sp, -4   # adjust stack pointer
   sw ra, 0(sp)      # save return address
   li a0, 3    # load test input N
   jal FACT    # call fact procedure
   lw ra, 0(sp)      # restore return address
   addi sp, sp, 4    # return stack pointer
   jr ra
wrap_end:
# ---------------------------------------------------------------
# Procedure FACT: compute N!
# param[in] a0 integer N
```

```
# return s0 the largest value
# ------------------------------------------------------------------------
FACT:
   addi sp, sp, -8 # allocate space for ra, a0 in stack
   sw ra, 4(sp)      # save ra register
   sw a0, 0(sp)      # save a0 register
   li t0, 2
   bge a0, t0, recursive
   li s0, 1    # return the result N!=1
   j done
recursive:
   addi a0, a0, -1 # adjust input argument
   jal FACT    # recursive call
   lw s1, 0(sp)      # load a0
   mul s0, s0, s1
done:
   lw ra, 4(sp)      # restore ra register
   lw a0, 0(sp)      # restore a0 register
   addi sp,sp,8      # restore stack pointer
   jr ra             # jump to caller
fact_end:
```

Kết quả:

STT	Vị trí	Thanh ghi	Giá trị thanh ghi	Giá trị vùng nhớ stack	Ghi chú
0		sp	0x7fffeffc		Trạng thái ban đầu của thanh ghi sp
1	main:	pc	0x00400000		
2	jal WARP	ra	0x00400004		Nhảy đến WARP, lưu địa chỉ trở về print vào ra
3	WARP:	pc	0x00400020		Bắt đầu procedure WARP
4	addi sp, sp, -4	sp	0x7fffeff8		Dịch stack pointer xuống 4 byte
5	sw ra,0(sp)			[sp] = 0x00400004	Lưu return address (0x00400004) vào stack
6	li a0, 3	a0	0x00000003		Gán a0 = 3 để tính 3!
7	jal FACT	ra	0x0040001C		Nhảy đến FACT, lưu địa chỉ trở về 0x0040001C vào ra
8	FACT:	pc	0x00400030		Bắt đầu procedure FACT
9	addi sp, sp, -8	sp	0x7fffeff0		Dịch stack pointer xuống 8 byte
10	sw ra, 4(sp)			[sp+4] = 0x0040001C	Lưu return address (0x0040001C) vào stack
11	sw a0, 0(sp)			[sp] = 0x00000003	Lưu giá trị a0 (3) vào stack
12	li t0, 2	t0	0x00000002		Gán t0 = 2 để so sánh
13	bge a0, t0, recursive	pc	0x00400048		Nhảy đến recursive vì 3 ≥ 2
14	recursive:	pc	0x00400048		Bắt đầu nhánh đệ quy
15	addi a0, a0, -1	a0	0x00000002		Giảm a0 từ 3 → 2
16	jal FACT	ra	0x0040004C		Gọi đệ quy FACT(2), lưu return address 0x0040004C vào ra
17	FACT:	pc	0x00400030		Bắt đầu FACT lần 2
18	addi sp, sp, -8	sp	0x7fffefe8		Dịch stack pointer
19	sw ra, 4(sp)			[sp+4] = 0x0040004C	Lưu return address
20	sw a0, 0(sp)			[sp] = 0x00000002	Lưu a0 = 2
21	bge a0, t0, recursive	pc	0x00400048		Nhảy đến recursive vì 2 ≥ 2
22	addi a0, a0, -1	a0	0x00000001		Giảm a0 từ 2 → 1
23	jal FACT	ra	0x0040004C		Gọi đệ quy FACT(1)
24	FACT:	pc	0x00400030		Bắt đầu FACT lần 3
25	addi sp, sp, -8	sp	0x7fffefe0		Dịch stack pointer
26	sw ra, 4(sp)			[sp+4] =0x0040004C	Lưu return address

27	sw a0, 0(sp)			[sp] = 0x00000001	Lưu a0 = 1
28	li t0, 2	t0	0x00000002		Gán t0 = 2
29	bge a0, t0, recursive	pc	0x00400044		Không nhảy vì 1 < 2
30	li s0, 1	s0	0x00000001		Gán s0 = 1 (trường hợp cơ sở)
31	j done	pc	0x00400058		Nhảy đến done
32	done:	pc	0x00400058		Bắt đầu khôi phục
33	lw ra, 4(sp)	ra	0x0040004C		Khôi phục return address
34	lw a0, 0(sp)	a0	0x00000001		Khôi phục a0
35	addi sp, sp, 8	sp	0x7fffefe8		Giải phóng stack
36	jr ra	pc	0x0040004C		Nhảy về địa chỉ 0x0040004C
37	lw s1,0(sp)	s1	0x00000002	[sp] = 0x00000002	Lấy giá trị a0 cũ (2)
38	mul s0, s0, s1	s0	0x00000002		Tính 2! = 2 × 1 = 2
39	done:	pc	0x00400058		Tiếp tục khôi phục
40	lw ra, 4(sp)	ra	0x0040004C		Khôi phục return address
41	lw a0, 0(sp)	a0	0x00000002		Khôi phục a0
42	addi sp,sp, 8	sp	0x7fffeff0		Giải phóng stack
43	jr ra	pc	0x0040004C		Nhảy về địa chỉ 0x0040004C
44	lw s1,0(sp)	s1	0x00000003	[sp] = 0x00000003	Lấy giá trị a0 cũ (3)
45	mul s0, s0, s1	s0	0x00000006		Tính 3! = 3 × 2 = 6
46	done:	pc	0x00400058		Tiếp tục khôi phục
47	lw ra, 4(sp)	ra	0x0040001C		Khôi phục return address
48	lw a0, 0(sp)	a0	0x00000003		Khôi phục a0
49	addi sp, sp, 8	sp	0x7fffeff8		Giải phóng stack
50	jr ra	pc	0x0040001C		Nhảy về WARP
51	WARP:	pc	0x0040001C		Tiếp tục WARP
52	lw ra, 0(sp)	ra	0x00400004	[sp] = 0x00400004	Khôi phục return address
53	addi sp, sp, 4	sp	0x7fffeffc		Giải phóng stack, trả về trạng thái ban đầu
54	jr ra	pc	0x00400004		Nhảy về print
55	print:	pc	0x00400004		Bắt đầu in kết quả
56	add a1, s0, zero	a1	0x00000006		Truyền kết quả 6 vào a1
57	li a7, 56	a7	0x00000038		Thiết lập syscall in số nguyên
58	la a0, message	a0	Địa chỉ message		Load địa chỉ chuỗi message
59	quit:	pc	0x00400018		Chuẩn bị kết thúc

ASSIGNMENT 5:

Đoạn mã:

```
.data
msg_max:    .asciz        # Chuỗi để in giá trị lớn nhất
msg_min:    .asciz        # Chuỗi để in giá trị nhỏ nhất
comma:      .asciz        # Chuỗi dấu phẩy
newline:    .asciz        # Chuỗi xuống dòng

.text
variable:                 # Nhãn khởi tạo các biến
    addi a0, zero, 2      # a0 = 2
    addi a1, zero, 6      # a1 = 6
    addi a2, zero, 1      # a2 = 1
    addi a3, zero, -3     # a3 = -3
    addi a4, zero, 5      # a4 = 5
    addi a5, zero, 3      # a5 = 3
    addi a6, zero, 2      # a6 = 2
    addi a7, zero, 9      # a7 = 9

main:
    addi sp, sp, -32      # Dịch con trỏ stack xuống 32 byte (8 words)
    sw a0, 0(sp)          # Lưu a0 vào stack[0]
    sw a1, 4(sp)          # Lưu a1 vào stack[4]
    sw a2, 8(sp)          # Lưu a2 vào stack[8]
    sw a3, 12(sp)         # Lưu a3 vào stack[12]
    sw a4, 16(sp)         # Lưu a4 vào stack[16]
    sw a5, 20(sp)         # Lưu a5 vào stack[20]
    sw a6, 24(sp)         # Lưu a6 vào stack[24]
    sw a7, 28(sp)         # Lưu a7 vào stack[28]
    li t6, 8              # t6 = 8 (số lượng phần tử)

    addi a0, sp, 0        # Truyền địa chỉ mảng vào a0
    jal find_max_min      # Gọi hàm tìm max/min

    li a7, 10             # Chuẩn bị syscall exit
    ecall                 # Kết thúc chương trình

find_max_min:
    lw s0, 0(a0)          # Khởi tạo max = phần tử đầu (s0)
    lw s2, 0(a0)          # Khởi tạo min = phần tử đầu (s2)
    li s1, 0              # Chỉ số của max (s1)
    li s3, 0              # Chỉ số của min (s3)

    li t0, 1              # Bộ đếm vòng lặp i = 1 (t0)
loop:
    bge t0, t6, done      # Nếu i >= số phần tử -> kết thúc
    add t1, t0, t0        # t1 = i*2
    add t1, t1, t1        # t1 = i*4 (offset phần tử thứ i)
    add t2, a0, t1        # t2 = địa chỉ phần tử thứ i
    lw t3, 0(t2)          # t3 = giá trị phần tử thứ i

    bge t3, s0, update_max # Nếu t3 >= max -> cập nhật max
```

```
    ble t3, s2, update_min # Nếu t3 <= min -> cập nhật min
    j next                 # Nhảy qua next

update_max:
    mv s0, t3              # Cập nhật giá trị max mới
    mv s1, t0              # Cập nhật chỉ số max mới
    j next                 # Tiếp tục vòng lặp

update_min:
    mv s2, t3              # Cập nhật giá trị min mới
    mv s3, t0              # Cập nhật chỉ số min mới

next:
    addi t0, t0, 1         # Tăng bộ đếm i++
    j loop                 # Lặp lại

done:
    jr ra                  # Trở về hàm gọi
```

Kết quả:

STT	Vị trí	Thanh ghi	Giá trị thanh ghi	Giá trị vùng nhớ stack	Ghi chú chi tiết
1	variable:	pc	0x00400000		Bắt đầu chương trình
2	addi a0, zero, 2	a0	0x00000002		Khởi tạo a0 = 2
3	addi a1, zero, 6	a1	0x00000006		Khởi tạo a1 = 6
4	addi a2, zero, 1	a2	0x00000001		Khởi tạo a2 = 1
5	addi a3, zero, -3	a3	0xFFFFFFFD		Khởi tạo a3 = -3
6	addi a4, zero, 5	a4	0x00000005		Khởi tạo a4 = 5
7	addi a5, zero, 3	a5	0x00000003		Khởi tạo a5 = 3
8	addi a6, zero, 2	a6	0x00000002		Khởi tạo a6 = 2
9	addi a7, zero, 9	a7	0x00000009		Khởi tạo a7 = 9
10	main:	pc	0x00400020		Bắt đầu hàm main
11	addi sp, sp, -32	sp	0x7fffefdc		Cấp phát 32 byte stack
12	sw a0, 0(sp)			[sp]=0x00000002	Lưu a0 vào stack
13	sw a1, 4(sp)			[sp+4]=0x00000006	Lưu a1 vào stack
14	sw a2, 8(sp)			[sp+8]=0x00000001	Lưu a2 vào stack
15	sw a3, 12(sp)			[sp+12]=0xFFFFFFFD	Lưu a3 vào stack
16	sw a4, 16(sp)			[sp+16]=0x00000005	Lưu a4 vào stack
17	sw a5, 20(sp)			[sp+20]=0x00000003	Lưu a5 vào stack
18	sw a6, 24(sp)			[sp+24]=0x00000002	Lưu a6 vào stack
19	sw a7, 28(sp)			[sp+28]=0x00000009	Lưu a7 vào stack
20	li t6, 8	t6	0x00000008		Gán t6 = 8 (số phần tử)

STT	Vị trí	Thanh ghi	Giá trị thanh ghi	Giá trị vùng nhớ stack	Ghi chú chi tiết
21	addi a0, sp, 0	a0	0x7fffefdc		Truyền địa chỉ mảng vào a0
22	jal find_max_min	ra	0x00400034		Gọi hàm find_max_min, lưu return address
23	find_max_min:	pc	0x00400040		Bắt đầu hàm find_max_min
24	lw s0, 0(a0)	s0	0x00000002		Khởi tạo max = phần tử đầu
25	lw s2, 0(a0)	s2	0x00000002		Khởi tạo min = phần tử đầu
26	li s1, 0	s1	0x00000000		Chỉ số max ban đầu = 0
27	li s3, 0	s3	0x00000000		Chỉ số min ban đầu = 0
28	li t0, 1	t0	0x00000001		Bộ đếm vòng lặp i = 1
29	loop:	pc	0x00400054		Bắt đầu vòng lặp
...	...	...	...	...	... (tiếp tục cho các vòng lặp)
40	done:	pc	0x00400078		Kết thúc hàm find_max_min
41	jr ra	pc	0x00400034		Trở về hàm main
42	li a7, 10	a7	0x0000000A		Chuẩn bị syscall exit
43	ecall				Kết thúc chương trình