Báo Cáo bài thực hành Lab 4
Môn: Tổ chức & Cấu trúc máy tính II

Nguyễn Trường Giang
MSSV: 24520011

Nguyễn Ngọc Hưng
MSSV: 24520610

Ngày 19 tháng 5 năm 2025

Bài 1: Mô phỏng các chương trình bên dưới và cho biết ý nghĩa các chương trình.

Ví dụ 1:

```
        .data
var1:   .word 23

        .text
__start:
    lw  $t0, var1
    li  $t1,  5
    sw  $t1, var1
```

Ý nghĩa: Gán giá trị biến var1 = 5

Ví dụ 2:

```
        .data
array1: .space 12

        .text
__start:
    la  $t0, array1
    li  $t1,  5
    sw  $t1,  ($t0)
    li  $t1, 13
    sw  $t1,  4($t0)
    li  $t1, -7
    sw  $t1,  8($t0)
```

array1: cấp phát 12 byte (tức 3 word)
array1[0] = 5
array1[1] = 13
array1[2] = -7

Ví dụ 3:

```
    li   $v0, 5
    syscall
```

$v0 = 5 (theo quy ước MARS, đây là mã dịch vụ ”read integer”)
Lệnh syscall: hệ thống sẽ đợi người dùng nhập vào một số nguyên vào $v0 (biến được gán trước đó).

Ví dụ 4:

```
        .data
string1: .asciiz "Print this.\n"

        .text
main:
    li   $v0, 4
    la   $a0, string1
    syscall
```

string1: chứa xâu ký tự ”Print this.\n”
$v0 = 4 (đây là mã dịch vụ ”print string”)
$a0 = địa chỉ của string1
syscall: với mã dịch vụ ”print string”, lệnh syscall sẽ gọi hàm in ra xâu và sẽ xuất dãy kí tự nằm tại địa chỉ $a0.

Bài 2. Thao tác với mảng

1. In ra cửa sổ I/O của MARS tất cả các phần tử của mảng array1 và array2
2. Gán các giá trị cho mảng array3 sao cho $array3[i] = array2[i] + array2[size2 - 1 - i]$
3. Người sử dụng nhập vào mảng thứ mấy và chỉ số phần tử cần lấy trong mảng đó, chương trình xuất ra phần tử tương ứng.

```
.data
array1: .word 5, 6, 7, 8, 1, 2, 3, 9, 10, 4
size1:  .word 10

array2: .byte 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16
size2:  .word 16

array3: .space 8       # Cap phat 8 byte, chua khoi gia tri
size3:  .word 8

# Chuoi de in newline va in prompt
newline:   .asciiz "\n"

promptArr: .asciiz "nhap mang (1-3): "
promptIdx: .asciiz "nhap chi so: "

.text
.globl main

main:
    # -------------------------------------
    # -------------------------------------
    # 1) In ra toan bo cac phan tu cua array1
    la   $t0, size1
    lw   $t2, 0($t0)     # $t2 = size1 = 10

    li   $t1, 0          # $t1 = i = 0
    la   $t3, array1     # $t3 = base address cua array1

print_array1_loop:
    beq  $t1, $t2, done_print_array1
    # Tinh address = base + i*4
    sll  $t4, $t1, 2     # $t4 = i << 2
    add  $t5, $t3, $t4   # $t5 = &array1[i] (pseudo code)
    lw   $t6, 0($t5)     # $t6 = array1[i]

    # In gia tri ($t6) ra man hinh
```

```
    add $a0, $t6, $zero
    li   $v0, 1 # ma "print integer"
    syscall

    # In mot dau space
    li   $a0, 32         # ASCII code = 32 = ' '
    li   $v0, 11 # print ki tu ASCII ra man hinh
    syscall

    addi $t1, $t1, 1     # i++
    j    print_array1_loop

done_print_array1: # ket thuc vong lap

    # In newline
    la   $a0, newline
    li   $v0, 4
    syscall

    # -------------------------------------
    # -------------------------------------
    # 2) In ra toan bo cac phan tu cua array2
    la   $t0, size2
    lw   $t2, 0($t0)     # $t2 = size2 = 16

    li   $t1, 0
    la   $t3, array2

print_array2_loop:
    beq  $t1, $t2, done_print_array2
    # Address = base + i
    add  $t5, $t3, $t1   # $t5 = &array2[i]
    lbu  $t6, 0($t5)     # $t6 = array2[i] # lenh co ban, load 1 byte (8 bit)

    # In $t6
    add $a0, $t6, $zero
    li   $v0, 1
    syscall

    # In space
    li   $a0, 32
    li   $v0, 11
    syscall

    addi $t1, $t1, 1
    j    print_array2_loop

done_print_array2:
    # In newline
    la   $a0, newline
    li   $v0, 4 # print string
    syscall

    # -------------------------------------
    # -------------------------------------
    # 3) Tinh array3[i] = array2[i] + array2[size2-1-i]
    la   $t0, size2
    lw   $t2, 0($t0)     # $t2 = size2 = 16
    la   $t0, size3
    lw   $t3, 0($t0)     # $t3 = size3 = 8

    li   $t1, 0
    la   $t4, array2
```

```
    la   $t6, array3

compute_array3_loop:
    beq  $t1, $t3, done_compute_array3

    # Lay array2[i]
    add  $t7, $t4, $t1   # $t7 = &array2[i]
    lbu  $t5, 0($t7)     # $t5 = array2[i]

    # Tinh j = size2 - 1 - i
    addi $t8, $t2, -1   # t8 = size2 - 1
    sub  $t8, $t8, $t1 # t8 = (size2-1) - i

    # Lay array2[j]
    add  $t8, $t8, $t4 # $t8 = &array2[j]
    lbu  $t7, 0($t8)   # $t7 = array2[j]

    # Cong
    add  $t5, $t5, $t7
    # Luu vao array3[i]
    add  $t7, $t6, $t1  # $t7 = &array3[i]
    sb   $t5, 0($t7)   # ghi 1 byte

    addi $t1, $t1, 1
    j    compute_array3_loop

done_compute_array3:
    # (Khong in array3 ra, vi de khong yeu cau)

    # ---------------------------------------
    # ---------------------------------------
    # 4) Cho nguoi dung nhap: so mang (1-3) va chi so, roi in phan tu tuong ung
        ↪ (khong co thong bao loi)
    # In prompt "Enter array number (1-3): "
    la   $a0, promptArr
    li   $v0, 4
    syscall

    # Doc so mang vao $t0
    li   $v0, 5 # read integer
    syscall
    add $t0, $v0, $zero          # $t0 = array_number

    # In prompt "Enter index: "
    la   $a0, promptIdx
    li   $v0, 4
    syscall

    # Doc index vao $t1
    li $v0, 5 # read integer
    syscall
    add $t1, $v0, $zero          # $t1 = index

    # So sanh va xu ly tung mang (neu khac 1,2,3 → exit)
    li   $t2, 1
    beq  $t0, $t2, handle_array1
    li   $t2, 2
    beq  $t0, $t2, handle_array2
    li   $t2, 3
    beq  $t0, $t2, handle_array3

    # Neu t0   1,2,3 thi nhay thang ra exit
    j    exit_program
```

```

handle_array1:
    # Tinh dia chi array1[index]
    # (gia su index hop le 0 <= index < size1)
    sll  $t3, $t1, 2       # $t3 = index << 2
    la   $t4, array1       # $t4 = base array1
    add  $t5, $t4, $t3     # $t5 = &array1[index]
    lw   $t6, 0($t5)       # $t6 = array1[index]
    j    print_result

handle_array2:
    la   $t4, array2       # $t4 = base array2
    add  $t5, $t4, $t1     # $t5 = &array2[index]
    lbu  $t6, 0($t5)       # $t6 = array2[index]
    j    print_result

handle_array3:
    la   $t4, array3       # $t4 = base array3
    add  $t5, $t4, $t1     # $t5 = &array3[index]
    lbu  $t6, 0($t5)       # $t6 = array3[index]
    j    print_result

print_result:
    # In ket qua ($t6) roi newline
    add $a0, $t6, $zero
    li   $v0, 1 #  print integer
    syscall

    la   $a0, newline
    li   $v0, 4 # print string
    syscall

exit_program:
    li   $v0, 10
    syscall
```

Giải thích một số câu lệnh "lạ".

- **lbu**: Load Byte Unsigned: tải một byte (8 bit) từ địa chỉ bộ nhớ lên thanh ghi đích, phần còn lại (tức là 24 bit cao) được tự động điền bằng 0 (không mở rộng dấu)
- **li**: Load Immediate (li $t, imm) là pseudo-instruction, dùng để nạp một giá trị nguyên vào thanh ghi.
- **la**: Load Address (la $t, label$) là pseudo-instruction, nạp địa chỉ của biên/nhãn.
- **syscall**: Là cơ chế để gọi system services (nhập xuất, quản lý bộ nhớ, thoát chương trình).
  - **$v0 = 1**: Print integer. (in một số nguyên từ thanh ghi $a0).
  - **$v0 = 4**: Print String. (in ra một string kết thúc bằng \0 từ địa chỉ trong $a0).
  - **$v0 = 5**: Read Integer. (đọc số nguyên từ bàn phím, lưu vào $v0).
  - **$v0 = 11**: Print Character. (in ký tự ASCII từ thanh ghi $a0).
  - **$v0 = 10**: Exit Program.

Bài 3

```
.data
newline:    .asciiz "\n"

promptArr: .asciiz "nhap n: "
promptIdx: .asciiz "nhap gia tri thu "
promptSpace: .asciiz ": "
```

```
7
8  .text
9
10     la $a0, promptArr
11     li $v0, 4
12     syscall
13
14     # doc n
15     li $v0, 5
16     syscall
17     add $s2, $v0, $zero # $s2 = n
18
19
20     add $t0, $zero, $zero
21
22     Loop:
23     beq $t0, $s2, Endloop
24
25         la $a0, promptIdx
26         li $v0, 4
27         syscall
28
29         add $a0, $zero, $t0
30         li $v0, 1
31         syscall
32
33         la $a0, promptSpace
34         li $v0, 4
35         syscall
36
37         # nhap a[i]
38         li $v0, 5
39         syscall
40         sll $t1, $t0, 2
41         add $t1, $t1, $s3
42         sw $v0, 0($t1)
43
44         addi $t0, $t0, 1
45     j Loop
46     Endloop:
47
48
49 # so sanh i < j
50     slt   $t0, $s0, $s1
51     beq   $t0, $zero, ELSE
52
53     # TH1: i < j  --> A[i] = i
54     sll   $t1, $s0, 2
55     add   $t1, $t1, $s3
56     sw    $s0, 0($t1)
57     j     END_IF
58
59 ELSE:
60     # TH2: i >= j  --> A[i] = j
61     sll   $t1, $s0, 2
62     add   $t1, $t1, $s3
63     sw    $s1, 0($t1)
64
65 END_IF:
```