ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA KỸ THUẬT MÁY TÍNH

TỔ CHỨC VÀ CẤU TRÚC MÁY TÍNH II

(IT012)

Sinh viên: Trần Nguyễn Thái Bình

MSSV: 23520161

Giảng viên hướng dẫn: Nguyễn Thành Nhân

IT012.O21.1 – Lab 04

MỤC LỤC

1. Lý thuyết

- Giảng viên hướng dẫn sinh viên về chương trình hợp ngữ MIPS dựa theo tài liệu: **Tổng quát về hợp ngữ và kiến trúc MIPS**

2. Thực hành

2.1. Bài 1

```
main:
    # nhap i
    li      $v0,        5
    syscall
    add     $s0,        $v0,    $zero

    # nhap j
    li      $v0,        5
    syscall
    add     $s1,        $v0,    $zero

    # nhap g
    li      $v0,        5
    syscall
    add     $t0,        $v0,    $zero

    # nhap h
    li      $v0,        5
    syscall
    add     $t1,        $v0,    $zero

    # so sanh
    beq     $s0,        $s1,    if
    sub     $s2,        $t0,    $t1

    j       continue

if:
    add     $s2,        $t0,    $t1

    # in ket qua
continue:
    li      $v0,        1
    add     $a0,        $s2,    $zero
    syscall
```

Chương trình mẫu:

1 7 5 3 2

Giải thích code:

Dòng	Chú thích	Ghi chú
3 → 5	Nhập i và lưu vào register **$s0**.	
8 → 10	Nhập j và lưu vào register **$s1**.	
12 → 15	Nhập g và lưu vào register **$t0**.	
18 → 20	Nhập h và lưu vào register **$t1**.	
23	So sánh **$s0** và **$s1.** **Nếu $s0 = $s1** thực hiện tiếp dòng 28, ngược lại thì thực hiện dòng tiếp theo.	So sánh i và j.
24	Lấy **$t0 - $t1** rồi lưu vào register **$s2**.	Lấy g – h rồi lưu vào **$s2**
28 → 29	Lấy **$t0 + $t1** rồi lưu vào register **$s2**.	Lấy g + h rồi lưu vào **$s2**
32 → 35	In kết quả đã tính toán.	

2.2.Bài 2

```
1.  .data
2.  i:      .word   1
3.  Sum:    .word   0
4.
5.  .text
6.  main:
7.
8.      # gan
9.      lw      $s0,    i
10.     lw      $s2,    Sum
11.
12.     # nhap N
13.     li      $v0,    5
14.     syscall
```

```
15.     add     $s1,    $v0,    $zero
16.
17. loop:       bgt     $s0,    $s1,    continue
18.     add     $s2,    $s2,    $s0
19.     addi    $s0,    $s0,    1
20.     j       loop
21.
22. continue:
23.     li      $v0,    1
24.     add     $a0,    $s2,    $zero
25.     syscall
```

Chương trình mẫu:

25 325

Giải thích code:

Dòng	Chú thích	Ghi chú
2	Lưu i = 1**.**	
3	Lưu Sum = 0**.**	
9	Lưu i vào register **$s0**.	
10	Lưu Sum vào register **$s2**.	
13 → 15	Nhập N và lưu vào register **$s1**.	
17 → 19	So sánh **$s0** và **$s1** Nếu **$s0 > $s1**, thực hiện tiếp dòng 22, ngược lại thực hiện dòng tiếp theo đến khi gặp dòng lệnh "**j loop**" thì quay lại dòng 17. Cộng **$s2 và $s0** rồi lưu vào **$s2**. Cộng **$s0 với 1** rồi lưu vào **$s0**.	Thực hiện cộng Sum và I trong vòng lặp với điều kiện i <= N, i++.
22 → 25	In ra **$s2**,	In ra Sum sau khi kết thúc vòng lặp.

3. Bài tập

3.1.Nhập vào một ký tự, xuất ra cửa sổ I/O của MARS theo từng yêu cầu

- Ký tự liền trước và liền sau của ký tự nhập vào

- Ký tự nhập vào chỉ được phép là ba loại: số, chữ thường và chữ hoa. Nếu ký tự nhập vào rơi vào một trong ba loại, xuất ra cửa sổ đó là loại nào; nếu ký tự nhập không rơi vào một trong ba loại trên, xuất ra thông báo “invalid type”

```
1.  .data
2.  string_truoc:       .asciiz "Ky tu truoc: "
3.  string_sau:         .asciiz "Ky tu sau: "
4.  string_invalid:     .asciiz "invalid type"
5.  string_xuong_dong:  .asciiz "\n"
6.
7.  .text
8.  main:
9.      # read character
10.     li      $v0,        12
11.     syscall
12.     add     $s0,        $v0,                $zero
13.     jal     xuong_dong
14.
15. la_so:
16.     li      $t0,        48
17.     blt     $s0,        $t0,                invalid
18.     li      $t0,        57
19.     ble     $s0,        $t0,                get
20.
21. la_chu_hoa:
22.     li      $t0,        65
23.     blt     $s0,        $t0,                invalid
24.     li      $t0,        90
25.     ble     $s0,        $t0,                get
26.
27. la_chu_thuong:
28.     li      $t0,        97
29.     blt     $s0,        $t0,                invalid
30.     li      $t0,        122
31.     ble     $s0,        $t0,                get
32.     j       invalid
33.
34. get:
35.     # get truoc va sau
36.     addi    $s1,        $s0,                -1
37.     addi    $s2,        $s0,                1
38.
```

```
valid:
    # in ra truoc
    li      $v0,        4
    la      $a0,        string_truoc
    syscall
    li      $v0,        11
    add     $a0,        $s1,                $zero
    syscall
    jal     xuong_dong
    # in ra sau
    li      $v0,        4
    la      $a0,        string_sau
    syscall
    li      $v0,        11
    add     $a0,        $s2,                $zero
    syscall
    jal     xuong_dong
    j       end

invalid:
    # invalid
    li      $v0,        4
    la      $a0,        string_invalid
    syscall
    j       end

xuong_dong:
    li      $v0,        4
    la      $a0,        string_xuong_dong
    syscall
    jr      $ra

end:
```

Chương trình mẫu

7 Ky tu truoc: 6 Ky tu sau: 8
u Ky tu truoc: t Ky tu sau: v

R Ky tu truoc: Q Ky tu sau: S
& invalid type

Giải thích code:

Dòng	Chú thích	Ghi chú
2 → 5	Lưu các biến **string_truoc** = "Ky tu truoc: " **string_sau** = "Ky tu sau: " **string_invalid** = "invalid type" **string_xuong_dong** = "\n"	
10 → 12	Nhập kí tự và lưu vào $s0.	
15 → 19	So sánh mã ascii của **$s0** với mã ascii 48 (số 0). Nếu nhỏ hơn 48 thì nhảy đến label **invalid.** Ngược lại tiếp tục thực hiện dòng tiếp theo, so sánh với mã ascii 57 (số 9), nếu nhỏ hơn hoặc bằng 57 thì nhảy đến label **get.**	Khối điều kiện kiểm tra kí tự nhập vào có phải là số hay không.
21 → 25	So sánh mã ascii của **$s0** với mã ascii 65 (“A”). Nếu nhỏ hơn 65 thì nhảy đến label **invalid.** Ngược lại tiếp tục thực hiện dòng tiếp theo, so sánh với mã ascii 90 (“Z”), nếu nhỏ hơn hoặc bằng 90 thì nhảy đến label **get**.	Khối điều kiện kiểm tra kí tự nhập vào có phải là chữ hoa hay không.
27 → 31	So sánh mã ascii của **$s0** với mã ascii 97 (“a”). Nếu nhỏ hơn 97 thì nhảy đến **label invalid.** Ngược lại tiếp tục thực hiện dòng tiếp theo, so sánh với mã ascii 122 (“z”), nếu nhỏ hơn hoặc bằng 122 thì nhảy đến label **get**.	Khối điều kiện kiểm tra kí tự nhập vào có phải là chữ thường hay không.
32	Nhảy đến label **invalid**.	
34 → 37	Lần lượt bớt và thêm 1 cho mã ascii của **$s0** rồi lưu vào **$s1 và $s2**.	Gán kí tự trước và sau kí tự đã nhập vào register **$s1, $s2**.

39 → 56	In ra màn hình như ví dụ sau: “b Ky tu truoc: a Ky tu sau: c”	Label **valid**
58 → 63	In ra thông báo “invalid type”.	Label **invalid**
65 → 69	In xuống dòng.	
71	Kết thúc.	

3.2.1. Nhập 2 số nguyên, in ra cửa sổ I/O của MARS theo từng yêu cầu

- Số lớn hơn
- Tổng, hiệu, tích, thương của 2 số

```
.data
 string_nhap_a:          .asciiz "Nhap a: "
 string_nhap_b:          .asciiz "Nhap b: "
 string_tong:            .asciiz "a + b = "
 string_hieu:            .asciiz "a - b = "
 string_nhan:            .asciiz "a * b = "
 string_khong_the_chia:  .asciiz "a khong the chia cho b!"
 string_chia:            .asciiz "a / b = "
 string_du:              .asciiz "du = "
 string_soLon:           .asciiz "So lon hon la: "
 string_xuong_dong:      .asciiz "\n"

 .text

 main:
     # nhap a
     la      $a0,        string_nhap_a
     li      $v0,        4
     syscall
     li      $v0,        5
     syscall
     add     $s0,        $v0,                    $zero
```

```
23.
24.     # nhap b
25.     la      $a0,        string_nhap_b
26.     li      $v0,        4
27.     syscall
28.     li      $v0,        5
29.     syscall
30.     add     $s1,        $v0,                    $zero
31.
32.  jal xuat_so_lon
33.  slt $t1, $s0, $s1
34.  beqz $t1, soLon
35.  add $a0, $s1, $zero
36.  li $v0, 1
37.  syscall
38.  jal xuong_dong
39.  j tong
40.
41. soLon:
42.     add $a0, $s0, $zero
43.     li $v0, 1
44.     syscall
45.     jal xuong_dong
46.
47. tong:
48.
49.     # xuat tinh tong
50.     la      $a0,        string_tong
51.     li      $v0,        4
52.     syscall
53.
54.     # xuat gia tri tinh tong
55.     add     $a0,        $s0,                    $s1
56.     li      $v0,        1
57.     syscall
58.     jal     xuong_dong
59.
60. hieu:
61.
62.     # xuat tinh hieu
63.     la      $a0,        string_hieu
64.     li      $v0,        4
65.     syscall
66.
```

```
    # xuat gia tri tinh hieu
    sub     $a0,        $s0,                $s1
    li      $v0,        1
    syscall
    jal     xuong_dong

nhan:

    # xuat tinh nhan
    la      $a0,        string_nhan
    li      $v0,        4
    syscall

    # xuat gia tri tinh nhan
    mult    $s0,        $s1
    mflo    $a0
    li      $v0,        1
    syscall
    jal     xuong_dong

chia:

    # kiem tra b == 0
    beq     $s1,        $zero,              khong_the_chia

    # xuat tinh chia
    la      $a0,        string_chia
    li      $v0,        4
    syscall

    # xuat gia tri thuong
    div     $s0,        $s1
    mflo    $a0
    li      $v0,        1
    syscall
    jal     xuong_dong

    # so du
    mfhi    $t0
    beq     $t0,        $zero,              end
    la      $a0,        string_du
    li      $v0,        4
    syscall
    add     $a0,        $t0,                $zero
```

```
111.      li      $v0,        1
112.      syscall
113.      j       end
114.
115.  khong_the_chia:
116.
117.      # xuat khong the chia
118.      la      $a0,        string_khong_the_chia
119.      li      $v0,        4
120.      syscall
121.      j       end
122.
123. xuat_so_lon:
124.
125.      li      $v0,        4
126.      la      $a0,        string_soLon
127.      syscall
128.      jr      $ra
129.
130.  xuong_dong:
131.      li      $v0,        4
132.      la      $a0,        string_xuong_dong
133.      syscall
134.      jr      $ra
135.
136.  end:
```

Chương trình mẫu:

Nhap a: 9 Nhap b: 3 So lon hon la: 9 a + b = 12 a - b = 6 a * b = 27 a / b = 3
Nhap b: 5 So lon hon la: 9 a + b = 14

a - b = 4 a * b = 45 a / b = 1 du = 4
Nhap a: 11 Nhap b: 0 So lon hon la: 11 a + b = 11 a - b = 11 a * b = 0 a khong the chia cho b!

Giải thích code:

Dòng	Chú thích	Ghi chú
2 → 11	Lưu các biến string cần thiết để xuất trong quá trình chạy chương trình.	
17→ 19	Xuất yêu cầu nhập a.	
20 → 22	Nhập a và lưu giá trị vào register **$s0**.	
25 → 27	Xuất yêu cầu nhập b.	
28 → 30	Nhập b và lưu giá trị vào register **$s1**.	
32	Xuất "Số lớn hơn là: ".	
33 → 45	So sánh **$s0** và **$s1**, sau đó xuất ra số lớn hơn.	So sánh a và b
50 → 52	Xuất tính tổng.	
55 → 57	Tính tổng của **$s0** và **$s1** lưu vào **$a0**. Sau đó xuất ra màn hình giá trị **$a0**.	a + b
58	Xuất xuống dòng.	
63 → 65	Xuất tính hiệu.	
68 → 70	Tính hiệu của **$s0** và **$s1** lưu vào **$a0**. Sau đó xuất ra màn hình giá trị **$a0**.	a – b
71	Xuất xuống dòng.	

76 → 78	Xuất tính nhân.	
81 → 84	Tính nhân *(tích)* của **$s0** và **$s1** lưu vào **$a0**. Sau đó xuất ra màn hình giá trị **$a0**.	a * b
85	Xuất xuống dòng.	
90	Kiểm tra xem **$s1** có bằng **$zero**? Nếu bằng thì jump đến label **khong_the_chia** và jump đến label **end** kết thúc chương trình.	b == 0 ?
93 → 95	Xuất tính chia.	
98 → 101	Tính chia *(thương)* **$s0** chia **$s1** và lưu kết quả vào **$a0**. Sau đó xuất ra màn hình giá tị **$a0**.	a / b
102	Xuất xuống dòng.	
105	Lưu số dư của kêt quả phép chia trên vào **$t0.**	
106	Kiểm tra **$t0** có bằng 0? Nếu bằng thì jump đến label **end** và kết thúc chương trình. Nếu không bằng thì tiếp tục.	Số dư == 0 ?
107→ 109	Xuất tính dư.	
110 → 113	Lưu giá trị **$t0** vào **$a0** và xuất ra màn hình giá trị **$a0**. Jump đến label **end** và kết thúc chương trình.	Xuất số dư.

3.2.2. Nhập 2 số thực, in ra cửa sổ I/O của MARS theo từng yêu cầu

- Số lớn hơn
- Tổng, hiệu, tích, thương của 2 số

```
1. .data
```

```
2.  string_nhap_a:          .asciiz "Nhap a: "
3.  string_nhap_b:          .asciiz "Nhap b: "
4.  string_tong:            .asciiz "a + b = "
5.  string_hieu:            .asciiz "a - b = "
6.  string_nhan:            .asciiz "a * b = "
7.  string_khong_the_chia:  .asciiz "a khong the chia cho b!"
8.  string_chia:            .asciiz "a / b = "
9.  string_xuong_dong:      .asciiz "\n"
10.
11. .text
12. main:
13.     # nhap a
14.     la      $a0,            string_nhap_a
15.     li      $v0,            4
16.     syscall
17.     li      $v0,            6
18.     syscall
19.     mov.s   $f1,            $f0
20.
21.     # nhap b
22.     la      $a0,            string_nhap_b
23.     li      $v0,            4
24.     syscall
25.     li      $v0,            6
26.     syscall
27.     mov.s   $f2,            $f0
28.
29. tong:
30.     # xuat tinh tong
31.     la      $a0,            string_tong
32.     li      $v0,            4
33.     syscall
34.     # xuat gia tri tinh tong
35.     add.s   $f12,           $f1,                    $f2
36.     li      $v0,            2
37.     syscall
38.     jal     xuong_dong
39.
40. hieu:
41.     # xuat tinh hieu
42.     la      $a0,            string_hieu
43.     li      $v0,            4
44.     syscall
45.     # xuat gia tri tinh hieu
```

```
    sub.s   $f12,           $f1,                    $f2
    li      $v0,            2
    syscall
    jal     xuong_dong

nhan:
    # xuat tinh nhan
    la      $a0,            string_nhan
    li      $v0,            4
    syscall
    # xuat gia tri tinh nhan
    mul.s   $f12,           $f1,                    $f2
    li      $v0,            2
    syscall
    jal     xuong_dong

chia:
    # kiem tra b == 0
    mtc1    $zero,          $f0
    c.eq.s  $f2,            $f0
    bc1t    khong_the_chia
    # xuat tinh chia
    la      $a0,            string_chia
    li      $v0,            4
    syscall
    # xuat gia tri thuong
    div.s   $f12,           $f1,                    $f2
    li      $v0,            2
    syscall
    j       end

khong_the_chia:
    # xuat khong the chia
    la      $a0,            string_khong_the_chia
    li      $v0,            4
    syscall
    j       end

xuong_dong:
    li      $v0,            4
    la      $a0,            string_xuong_dong
    syscall
    jr      $ra

```

```
90. end:
```

Chương trình mẫu:

Nhap a: 5.29 Nhap b: 4.11 a + b = 9.4 a - b = 1.1799998 a * b = 21.741901 a / b = 1.2871046
Nhap a: 11.11 Nhap b: 0 a + b = 11.11 a - b = 11.11 a * b = 0.0 a khong the chia cho b!

Giải thích code:

Dòng	Chú thích	Ghi chú
2 → 9	Lưu các biến string cần thiết để xuất trong quá trình chạy chương trình.	
14 → 16	Xuất yêu cầu nhập a.	
17 → 19	Nhập a và lưu giá trị vào register **$f1**.	
22 → 24	Xuất yêu cầu nhập b.	
25 → 27	Nhập b và lưu giá trị vào register **$f2**.	
31 → 33	Xuất tính tổng.	
35 → 37	Tính tổng của **$f1** và **$f2** lưu vào **$f12**. Sau đó xuất ra màn hình giá trị **$f12**.	a + b
38	Xuất xuống dòng.	
42 → 44	Xuất tính hiệu.	
46 → 48	Tính hiệu của **$f1** và **$f2** lưu vào **$12**. Sau đó xuất ra màn hình giá trị **$f12**.	a – b

49	Xuất xuống dòng.	
53 → 55	Xuất tính nhân.	
57 → 59	Tính nhân *(tích)* của **$f1** và **$f2** lưu vào **$f12**. Sau đó xuất ra màn hình giá trị **$f12**.	a * b
60	Xuất xuống dòng.	
64	Gán giá trị 0 của **$zero** vào register **$f0**	
65	So sánh giá trị của **$f2** có bằng 0 *(giá trị của $f0)*? Đúng sẽ gán giá trị 1 cho flag, sai sẽ gán giá trị 0.	b == 0 ?
66	Nếu flag bằng 1 thì jump đến label **khong_the_chia**. Sau đó jump đến label **end** và kết thúc chương trình. Nếu flag bằng 0 thì tiếp tục chương trình.	
68 → 70	Xuất tính chia.	
72 → 74	Tính chia *(thương)* **$f1** chia **$f2** và lưu kết quả vào **$f12**. Sau đó xuất ra màn hình giá tị **$f12**.	a / b
75	Jump đến label **end** và kết thúc chương trình.	

4. Bài tập về nhà

4.1. Nhập 1 chuỗi, xuất ra chuỗi đảo ngược

```
1. .data
2. string_nhap:    .asciiz "Nhap chuoi: "
3. string_xuat:    .asciiz "Chuoi dao nguoc: "
4. string:         .space  100
5.
6. .text
7. main:
8. nhap:
```

```
 9.     li      $v0,        4
10.     la      $a0,        string_nhap
11.     syscall
12.
13.     li      $v0,        8
14.     la      $a0,        string
15.     li      $a1,        100
16.     syscall
17.     la      $t1,        string
18.     add     $t1,        $t1,            $a1
19.     addi    $t2,        $a1,            0
20.
21. xuat:
22.     li      $v0,        4
23.     la      $a0,        string_xuat
24.     syscall
25.
26. xuat_nguoc:
27.     slti    $s1,        $t2,            0
28.     bnez    $s1,        end
29.     lb      $a0,        0($t1)
30.     li      $v0,        11
31.     syscall
32.     addi    $t2,        $t2,            -1
33.     addi    $t1,        $t1,            -1
34.     j       xuat_nguoc
35. end:
```

Chương trình mẫu:

Nhap chuoi: Xin Chao toi ten la Kev Chuoi dao nguoc: veK al net iot oahC niX

Giải thích code:

Dòng	Chú thích	Ghi chú
2 → 3	Lưu các biến string cần thiết để xuất trong quá trình chạy chương trình.	
4	Label **string** là kiểu **space** với kích thước 100.	= 100 bytes
9 → 11	Xuất yêu cầu nhập chuỗi.	

13 → 19	Nhập chuỗi độ dài tối đa = 100, **$a1 = 100** Lưu địa chỉ kí tự đầu tiên của chuỗi vào **$t1** Lưu địa chỉ kí tự cuối của chuỗi vào **$t1** bằng cách cho **$t1 + $a1.** Lưu **$t2 = $a1.**	
21 → 24	Xuất chuỗi đã nhập.	
26 → 27	So sánh **$t2** với **0** rồi lưu vào **$s1 ($s1 = 1** nếu **$t2 < 0** và ngược lại **$s1 = 0).**	Thực hiện in ngược chuỗi đã nhập với điều kiện **$t2 >= 0, $t2 --.**
28	Lệnh nhảy tới **Label end** nếu giá trị trong **$s1** khác 0, tức là nếu **$t2** nhỏ hơn 0.	
29 → 31	In kí tự tại địa chỉ **$t1.**	
32 → 33	Giảm **$t2** và **$t1.**	
34	Nhảy về thực hiện lại từ dòng 26.	

4.2. Nhập 1 mảng số nguyên và sắp xếp theo thứ tự tăng dần