

IT3280

THỰC HÀNH KIẾN TRÚC MÁY TÍNH

BÁO CÁO FINAL-PROJECT

Giảng viên hướng dẫn:

TS. Lê Xuân THÀNH

Sinh viên:

Nguyễn Trung THÀNH

20176874

Ngày 21 tháng 6 năm 2020

Mục lục

1	Tổng Quan	2
1.1	Đề tài được phân công	2
1.2	Công cụ sử dụng	2
1.3	Source Code	2
1.4	Các lệnh thực hiện	3
2	Project 7 – Chương trình kiểm tra cú pháp lệnh MIPS	4
2.1	Phân tích cách thức thực hiện	4
2.2	Ý nghĩa của các thanh ghi được sử dụng	4
2.3	Source code	5
2.4	Kết quả chạy chương trình	18

1 Tổng Quan

1.1 Đề tài được phân công

1. Project 7 - Chương trình kiểm tra cú pháp lệnh MIPS

Trình biên dịch của bộ xử lý MIPS sẽ tiến hành kiểm tra cú pháp các lệnh hợp ngữ trong mã nguồn, xem có phù hợp về cú pháp hay không, rồi mới tiến hành dịch các lệnh ra mã máy. Hãy viết một chương trình kiểm tra cú pháp của 1 lệnh hợp ngữ MIPS bất kỳ (không làm với lệnh giả) như sau:

- Nhập vào từ bàn phím một dòng lệnh hợp ngữ. Ví dụ: beq \$s1, 32, \$t4.
 - Kiểm tra xem mã Opcode có đúng hay không? Trong ví dụ trên, opcode là beq là hợp lệ thì hiện thông báo "opcode: beq, hợp lệ".
 - Kiểm tra xem tên các toán hạng phía sau có hợp lệ hay không? Trong ví dụ tên, toán hạng \$s1 là hợp lệ, 31 là không hợp lệ, \$t4 thì khỏi phải kiểm tra nữa vì toán hạng trước đã bị sai rồi.
 - Cho biết lệnh hợp ngữ đó cần bao nhiêu chu kỳ thì mới thực hiện xong.
- Gợi ý: nên xây dựng một cấu trúc chứa khuôn dạng của từng lệnh với tên lệnh, kiểu của toán hạng 1, kiểu của toán hạng 2, kiểu của toán hạng 3, số chu kỳ thực hiện.

1.2 Công cụ sử dụng

Mars4_5

1.3 Source Code

<https://github.com/thanhff/Computer-Architecture-Lab/blob/master/Final-Project/>

1.4 Các lệnh thực hiện

Common MIPS instructions.

Notes: *op, funct, rd, rs, rt, imm, address, shamt* refer to fields in the instruction format. The program counter PC is assumed to point to the next instruction (usually 4 + the address of the current instruction). M is the byte-addressed main memory.

Assembly instruction	Instr. format	<i>op</i> <i>op/funct</i>	Meaning	Comments
<code>add \$rd, \$rs, \$rt</code>	R	0/32	$\$rd = \$rs + \$rt$	Add contents of two registers
<code>sub \$rd, \$rs, \$rt</code>	R	0/34	$\$rd = \$rs - \$rt$	Subtract contents of two registers
<code>addi \$rt, \$rs, imm</code>	I	8	$\$rt = \$rs + imm$	Add signed constant
<code>addu \$rd, \$rs, \$rt</code>	R	0/33	$\$rd = \$rs + \$rt$	Unsigned, no overflow
<code>subu \$rd, \$rs, \$rt</code>	R	0/35	$\$rd = \$rs - \$rt$	Unsigned, no overflow
<code>addiu \$rt, \$rs, imm</code>	I	9	$\$rt = \$rs + imm$	Unsigned, no overflow
<code>mfc0 \$rt, \$rd</code>	R	16	$\$rt = \rd	<i>rd</i> = coprocessor register (e.g. epc, cause, status)
<code>mult \$rs, \$rt</code>	R	0/24	$Hi, Lo = \$rs * \rt	64 bit signed product in Hi and Lo
<code>multu \$rs, \$rt</code>	R	0/25	$Hi, Lo = \$rs * \rt	64 bit unsigned product in Hi and Lo
<code>div \$rs, \$rt</code>	R	0/26	$Lo = \$rs / \$rt, Hi = \$rs \text{ mod } \rt	
<code>divu \$rs, \$rt</code>	R	0/27	$Lo = \$rs / \$rt, Hi = \$rs \text{ mod } \rt (unsigned)	
<code>mfhi \$rd</code>	R	0/16	$\$rd = Hi$	Get value of Hi
<code>mflo \$rd</code>	R	0/18	$\$rd = Lo$	Get value of Lo
<code>and \$rd, \$rs, \$rt</code>	R	0/36	$\$rd = \$rs \& \$rt$	Logical AND
<code>or \$rd, \$rs, \$rt</code>	R	0/37	$\$rd = \$rs \$rt$	Logical OR
<code>andi \$rt, \$rs, imm</code>	I	12	$\$rt = \$rs \& imm$	Logical AND, unsigned constant
<code>ori \$rt, \$rs, imm</code>	I	13	$\$rt = \$rs imm$	Logical OR, unsigned constant
<code>sll \$rd, \$rs, shamt</code>	R	0/0	$\$rd = \$rs \ll shamt$	Shift left logical (shift in zeros)
<code>srl \$rd, \$rs, shamt</code>	R	0/2	$\$rd = \$rs \gg shamt$	Shift right logical (shift in zeros)
<code>lw \$rt, imm(\$rs)</code>	I	35	$\$rt = M[\$rs + imm]$	Load word from memory
<code>sw \$rt, imm(\$rs)</code>	I	43	$M[\$rs + imm] = \rt	Store word in memory
<code>lbu \$rt, imm(\$rs)</code>	I	37	$\$rt = M[\$rs + imm]$	Load a single byte, set bits 8-31 of <i>\$rt</i> to zero
<code>sb \$rt, imm(\$rs)</code>	I	41	$M[\$rs + imm] = \rt	Store byte (bits 0-7 of <i>\$rt</i>) in memory
<code>lui \$rt, imm</code>	I	15	$\$rt = imm * 2^{16}$	Load constant in bits 16-31 of register <i>\$rt</i>
<code>beq \$rs, \$rt, imm</code>	I	4	if($\$rs == \rt) PC = PC + <i>imm</i> (PC always points to next instruction)	
<code>bne \$rs, \$rt, imm</code>	I	5	if($\$rs \neq \rt) PC = PC + <i>imm</i> (PC always points to next instruction)	
<code>slt \$rd, \$rs, \$rt</code>	R	0/42	if($\$rs < \rt) $\$rd = 1$; else $\$rd = 0$	
<code>slti \$rt, \$rs, imm</code>	I	10	if($\$rs < imm$) $\$rt = 1$; else $\$rt = 0$	
<code>sltu \$rd, \$rs, \$rt</code>	R	0/43	if($\$rs < \rt) $\$rd = 1$; else $\$rd = 0$ (unsigned numbers)	
<code>sltiu \$rt, \$rs, imm</code>	I	11	if($\$rs < imm$) $\$rt = 1$; else $\$rt = 0$ (unsigned numbers)	
<code>j destination</code>	J	2	PC = <i>address</i> *4	Jump to <i>destination</i> , <i>address</i> = <i>destination</i> /4
<code>jal destination</code>	J	3	$\$ra = PC$; PC = <i>address</i> *4 (Jump and link, <i>address</i> = <i>destination</i> /4)	
<code>jr \$rs</code>	R	0/8	PC = <i>\$rs</i>	Jump to address stored in register <i>\$rs</i>

2 Project 7 – Chương trình kiểm tra cú pháp lệnh MIPS

2.1 Phân tích cách thức thực hiện

Phân tích đề bài:

- Đầu vào: Câu lệnh chương trình MIPS.
- Đầu ra: Kiểm tra xem lệnh đã nhập vào có đúng lệnh MIPS hay không?

Ý tưởng:

- Xây dựng 1 Library có chứa cấu trúc của các lệnh hợp ngữ. (Cấu trúc của Library bao gồm: opcode - toán hạng - chu kỳ lệnh).
- Các ký hiệu trong Library: 1 - thanh ghi; 2 - hằng số nguyên; 3 - định danh; 4 - dành cho các lệnh lw, sw,... có cấu trúc ví dụ như imm(\$rs); 0 - không có.
- Tạo các khu vực chứa "character", "number", "register" để thực hiện kiểm tra.
- Duyệt câu lệnh nhập vào từ trái sang phải. Nếu opcode đúng, duyệt tiếp tới các toán hạng. Nếu Opcode sai thì thông báo sai (tương tự với các toán hạng cũng duyệt dần như vậy).

Cách thức thực hiện:

- **Bước 1:** Menu thực hiện nhập lệnh hoặc thoát.
- @input: Nhập lệnh MIPS vào từ bàn phím.
- **Bước 2:** Kiểm tra câu lệnh
 1. Kiểm tra Opcode (add, and, or,...).
 2. Sau khi kiểm tra Opcode kiểm tra dần tới các toán hạng. Nếu toán hạng đó là '0' trong Library thì thực hiện kiểm tra đằng sau đó có thừa ký tự gì không.
 3. Kiểm tra giữa 2 toán hạng cần phải có dấu ','.
 4. Khi kiểm tra các toán hạng cần xem trong Library toán hạng đó là thanh ghi, số nguyên, định danh, hay imm(\$rs). Rồi thực hiện đi đến so sánh từng giá trị này.
- **Bước 3:** Kiểm duyệt thành công in thông báo lệnh đúng cấu trúc và ngược lại. Sau đó quay trở lại Menu.

2.2 Ý nghĩa của các thanh ghi được sử dụng

- \$s7: lưu index của câu lệnh nhập vào.
- \$s6: lưu index của Opcode.
- \$s5: lưu vị trí Opcode trong Library.
- \$s4: lưu các toán hạng
- \$s3: lưu vị trí của toán hạng trong Library.
- \$s2: Lưu 0 hoặc 1 để bật hoặc tắt kiểm tra theo dạng imm(\$rs).

2.3 Source code

```
1 #-----
2 # Author: Nguyen Trung Thanh - 20176874
3 # Project 7: Chuong trinh kiem tra cu phap lenh Mips
4 #-----
5
6 .data
7 menu_mess:      .asciiiz "\n>>>>>>>>>MENU<<<<<<<<<\n1. Kiem tra cu
8   phap lenh\n2. Thoat \nChon: "
9 menu_error_mess: .asciiiz "\nNhap sai, vui long nhap lai!\n"
10 input_mess:      .asciiiz "\nNhap vao lenh Mips: "
11
12 opcode_mess:     .asciiiz "Opcode: "
13 toanHang_mess:   .asciiiz "Toan hang: "
14 hopLe_mess:      .asciiiz " - hop le.\n"
15 error_mess:      .asciiiz "\nLenh hop ngu khong hop le, sai khuon dang
16   lenh !\n"
17 completed_mess:  .asciiiz "\nLenh hop ngu chinh xac !\n"
18 chuKy_mess:      .asciiiz "So chu ky cua lenh la: "
19
20 command:         .space 100 # Luu cau lenh
21 opcode:          .space 30 # Luu ma lenh, vi du: add, and,...
22 ident:           .space 30 # nhan | hoac number
23 token:           .space 30 # cac thanh ghi, vi du: $zero, $at,...
24
25 # Cau truc cua library:
26 # opcode (6 byte) - operation - chu ky lenh
27 # Trong so luong operation: 1 - thanh ghi; 2 - hang so nguyen; 3 -
28   dinh danh (ident); 4 - imm($rs); 0 - khong co
29 library: .asciiiz "add***1111;sub***1111;addi**1121;addu**1111;addiu
30   *1121;subu**1111;mfc0**1101;mult**1101;multu*1101;div***1101;mfhi
31   **1001;mflo**1001;and***1111;or****1111;andi**1121;ori***1121;sll
32   ***1121;srl***1121;lw****1401;sw****1401;lb***1401;sb****1401;lui
33   ***1201;beq***1132;bne***1132;slt***1111;slti**1121;sltiu*1121;j
34   *****3001;jal***3001;jr****1001;nop***0001"
35 numberGroup:     .asciiiz "0123456789-"
36 characterGroup:  .asciiiz "0123456789
37   qwertyuiopasdfghjklmnbvcxzQWERTYUIOPASDFGHJKLZXCVBNM_"
38 # Moi thanh ghi cach nhau 6 byte
39 tokenRegisters:  .asciiiz "$zero $at      $v0      $v1      $a0      $a1      $a2
40   $a3      $t0      $t1      $t2      $t3      $t4      $t5      $t6      $t7      $s0      $s1
41   $s2      $s3      $s4      $s5      $s6      $s7      $t8      $t9      $k0      $k1      $gp
42   $sp      $fp      $ra      $0       $1       $2       $3       $4       $5       $7       $8
43   $9       $10      $11      $12      $13      $14      $15      $16      $17      $18      $19
44   $20      $21      $22      $21      $22      $23      $24      $25      $26      $27      $28
45   $29      $30      $31      "
46
47 .text
48
49 main:
50 # >>>>>>>>> MENU <<<<<<<<<
```

```

36 m_menu_start:
37     li $v0, 4
38     la $a0, menu_mess
39     syscall
40
41     # Read number input menu
42     li $v0, 5
43     syscall
44
45     beq $v0, 2, end_main      # 2: ket thuc
46     beq $v0, 1, m_menu_end    # 1: thuc hien kiem tra
47
48     li $v0, 4
49     la $a0, menu_error_mess   # Nhap sai
50     syscall
51
52     j m_menu_start
53 m_menu_end:
54
55     # >>>>>>>>> READ INPUT <<<<<<<<<<
56 m_input:
57     jal input
58     nop
59
60     # >>>>>>>>> START CHECK <<<<<<<<<<
61
62 m_check:
63     jal check
64     nop
65
66     j m_menu_start
67
68 end_main:
69     li $v0, 10
70     syscall
71
72     #-----
73     # 1. @input: Nhap vao lenh Mips tu ban phim
74     #-----
75 input:
76     li $v0, 4
77     la $a0, input_mess
78     syscall
79
80     li $v0, 8
81     la $a0, command
82     li $a1, 100
83     syscall
84
85     jr $ra
86
87     #-----

```

```

88 # 2. @check: Kiem tra cau lenh
89 # - Buoc 1: Kiem tra opcode (add, and, or,...) ten lenh
90 # - Buoc 2: Kiem tra Operand lan luot cac operand (Toan hang)
91 #-----
92 check:
93     # Luu $ra de tro ve main
94     addi $sp, $sp, -4
95     sw    $ra, 0($sp)
96
97     addi $s7, $zero, 0    # Thanh ghi $s7 luu index cua command
98
99     # START CHECK OPCODE
100    jal check_opcode
101    nop
102
103    # START CHECK OPERAND 1
104    li    $s3, 6          # Vi tri operand trong Library
105    jal check_operand
106    nop
107
108    # START CHECK OPERAND 2    # Neu khong co dau ',' ngan cach giua
109    operand_1 va operand_2 => FALSE
110    li    $s3, 7          # Vi tri operand trong Library
111    add $t0, $s5, $s3
112    lb    $t0, 0($t0)
113    beq $t0, 48, check_none    # Kiem tra neu operand = 0 -> ket thuc; ky
114    tu 0 trong ASCII
115
116    la    $a0, command
117    add $t0, $a0, $s7    # tro toi vi tri tiep tục của command
118    lb    $t1, 0($t0)
119    bne $t1, 44, not_found    # Dau ','
120    add $s7, $s7, 1
121
122    jal check_operand
123    nop
124
125    # START CHECK OPERAND 3    # Neu khong co dau ',' ngan cach giua
126    operand_1 va operand_2 => FALSE
127    li    $s3, 8          # Vi tri operand trong Library
128    add $t0, $s5, $s3
129    lb    $t0, 0($t0)
130    beq $t0, 48, check_none    # Kiem tra neu operand = 0 -> ket thuc; ky
131    tu 0 trong ASCII
132
133    la    $a0, command
134    add $t0, $a0, $s7    # tro toi vi tri tiep tục của command
135    lb    $t1, 0($t0)
136    bne $t1, 44, not_found    # Dau ','
137    add $s7, $s7, 1
138
139    jal check_operand

```



```

136 nop
137
138 # KIEM TRA KY TU THUA
139 j check_none
140
141 # Tra lai $ra de tro ve main
142 lw $ra, 0($sp)
143 addi $sp, $sp, 4
144 jr $ra
145
146 #-----
147 # 2.1 @check_opcode: Kiem tra cau lenh
148 # - Buoc 1: Lay cac opcode trong command da nhap
149 # - Buoc 2: So sanh voi trong bo tu dien xem co opcode do khong
150 # - Neu khong co ket thuc va quay lai menu
151 # - Meu co, luu lai dia chi opcode trong library va tiep tục kiem
    tra
152 # $a0: command
153 # $a1: opcode
154 # $s7: index of command
155 # $t9: index of opcode
156 #-----
157 check_opcode:
158     la $a0, command          # Dia chi cua command
159     la $a1, opcode           # Dia chi cua opcode
160     li $t0, 0
161
162 remove_space_command:        # Xoa cac dau cach phia truoc lenh
163     add $t1, $a0, $t0
164     lb $t2, 0($t1)
165     bne $t2, 32, end_remove_space_command    # Neu khong phai ' ' -> Ket
        thuc
166     addi $t0, $t0, 1
167     j remove_space_command
168 end_remove_space_command:
169
170     li $t9, 0                # index for opcode
171     li $s6, 0                # so luong cac ki tu cua opcode = 0
172 read_opcode:
173     add $t1, $a0, $t0        # Dich bit cua command
174     add $t2, $a1, $t9        # Dich bit cua opcode
175     lb $t3, 0($t1)
176
177     beq $t3, 32, read_opcode_done    # Neu co dau cach ' ' ket thuc
        read opcode
178     beq $t3, 10, read_opcode_done    # Neu dau '\n' ket thuc read
        opcode
179     beq $t3, 0, read_opcode_done     # Ket thuc chuoì
180
181     sb $t3, 0($t2)
182     addi $t9, $t9, 1
183     addi $t0, $t0, 1

```

```

184     j read_opcode
185 read_opcode_done:
186
187     addi $s6, $t9, 0           # $s6: So luong ki tu cua opcode
188     add $s7, $s7, $t0         # luu index cua command
189     la $a2, library
190     li $t0, -11
191
192 check_opcode_inlib:
193     addi $t0, $t0, 11         # Buoc nay bang 10 de nay den tung
194     Instruction
195     li $t1, 0                 # i = 0
196     li $t2, 0                 # j = 0
197     add $t1, $t1, $t0         # Cong buoc nay
198
199 compare_opcode:
200     add $t3, $a2, $t1         # t3 tro thanh vi tri tro den dau cua tung
201     Instruction
202     lb $t4, 0($t3)
203     beq $t4, 0, not_found
204     beq $t4, 42, check_len_opcode # Neu gap ky tu '*' => Kiem tra do
205     dai
206     add $t5, $a1, $t2         # Load opcode
207     lb $t6, 0($t5)
208     bne $t4, $t6, check_opcode_inlib # So sanh 2 ki tu, neu khong
209     bang nhau thi tinh den Instruction tiep theo.
210     addi $t1, $t1, 1         # i = i + 1
211     addi $t2, $t2, 1         # j = j + 1
212     j compare_opcode
213 check_len_opcode:
214     bne $t2, $s6, check_opcode_inlib
215 end_check_opcode_inlib:
216
217     add $s5, $t0, $a2         # Luu lai vi tri Instruction trong Library.
218
219     # >>>>>>>> In thong tin ra man hinh <<<<<<<<<
220     li $v0, 4
221     la $a0, opcode_mess
222     syscall
223
224     la $a3, opcode
225     li $t0, 0
226 print_opcode:
227     beq $t0, $t9, end_print_opcode
228     add $t1, $a3, $t0
229     lb $t2, 0($t1)
230     li $v0, 11
231     add $a0, $t2, $zero
232     syscall
233     addi $t0, $t0, 1
234     j print_opcode
235 end_print_opcode:

```

```

232
233     li $v0, 4
234     la $a0, hopLe_mess
235     syscall
236
237     jr $ra
238
239
240 #-----
241 # 2.2 @check_operand:
242 # a0: command
243 # s7: Luu index cua command
244 # s5: vi tri cua instruction trong library
245 #-----
246
247 check_operand:
248     # Luu $ra de tro ve check_operand
249     addi $sp, $sp, -4
250     sw    $ra, 0($sp)
251
252     add $t9, $s5, $s3      # Tro toi operand trong Library
253     lb   $t9, 0($t9)
254     addi $t9, $t9, -48     # Char -> Number
255
256     la   $a0, command
257     add  $t0, $a0, $s7
258
259     li $t1, 0             # i = 0
260     space_remove:        # Xoa cac khoang trang thua
261         add $t2, $t0, $t1
262         lb   $t2, 0($t2)   # Lay ky tu tiep theo
263         bne $t2, 32, end_space_remove # Ky tu ' '
264         addi $t1, $t1, 1   # i = i + 1
265         j   space_remove
266     end_space_remove:
267
268     add $s7, $s7, $t1     # Cap nhat lai index command
269
270     li $s2, 0             # Tat kich hoat check number_register
271     li $t8, 0             # Khong co
272     beq $t8, $t9, check_none
273     li $t8, 1             # Thanh ghi
274     beq $t8, $t9, go_register
275     li $t8, 2             # So hang nguyen
276     beq $t8, $t9, go_number
277     li $t8, 3             # Ident
278     beq $t8, $t9, go_ident
279     li $t8, 4             # Check number & register
280     beq $t8, $t9, go_number_register
281
282 end_check_operand:
283     # Tra lai $ra de tro ve check_operand

```

```

284 lw    $ra, 0($sp)
285 addi $sp, $sp, 4
286 jr    $ra
287
288 #-----
289 # jal toi cac ham check de kiem tra
290 #-----
291 go_register:      # Check register
292     jal check_register
293     nop
294     j end_check_operand
295
296 go_number:        # Check number
297     la $a2, numberGroup
298     jal check_ident
299     nop
300     j end_check_operand
301
302 go_ident:         # Check Ident
303     la $a2, characterGroup
304     jal check_ident
305     nop
306     j end_check_operand
307
308 go_number_register: # Check number-register
309     jal check_number_register
310     nop
311     j end_check_operand
312
313 #-----
314 # @check_none: Kiem tra xem con ky tu nao o cuoi khong
315 #-----
316 check_none:
317     la $a0, command
318     add $t0, $a0, $s7
319
320     lb $t1, 0($t0)
321
322     beq $t1, 10, none_ok # Ky tu '\n'
323     beq $t1, 0, none_ok # Ket thuc chuoai
324
325     j not_found
326
327 none_ok:
328     li $v0, 4
329     la $a0, chuKy_mess
330     syscall
331
332     li $s3, 9          # Vi tri operand trong Library
333     add $t0, $s5, $s3
334     lb $t0, 0($t0)
335

```

```

336     li $v0, 11
337     add $a0, $t0, $zero
338     syscall
339
340     li $v0, 4
341     la $a0, completed_mess
342     syscall
343     j m_menu_start
344
345     #-----
346     # @check_register: Kiem tra xem register co hop le hay khong
347     # a0: command (vi tri luu command)
348     # a1: token (vi tri luu thanh ghi)
349     # a2: tokenRegisters
350     # s7: Luu index cua command
351     # $t9: index cua token
352     #-----
353
354 check_register:
355     la $a0, command
356     la $a1, token
357     la $a2, tokenRegisters
358     add $t0, $a0, $s7          # Tro den vi tri cac instruction
359
360     li $t1, 0                 # i = 0
361     li $t9, 0                 # index cua token
362
363 read_token_register:
364     add $t2, $t0, $t1          # command
365     add $t3, $a1, $t1          # token
366     lb $t4, 0($t2)
367
368     beq $t4, 41, end_read_token    # Gap ky tu ')'
369     beq $t4, 44, end_read_token    # Gap ky tu ', '
370     beq $t4, 10, end_read_token    # Gap ky tu '\n'
371     beq $t4, 0, end_read_token     # Ket thuc
372
373     addi $t1, $t1, 1
374     beq $t4, 32, read_token_register    # Neu gap dau ' ' thi tiep tuc
375
376     sb $t4, 0($t3)
377     addi $t9, $t9, 1
378     j read_token_register
379
380 end_read_token:
381     add $s7, $s7, $t1          # Cap nhat lai gia tri index
382
383     li $t0, -6
384 compare_token_register:
385     addi $t0, $t0, 6           # Buoc nhay bang 6 de nhay den tung Register
386
387     li $t1, 0                 # i = 0

```

```

388 li $t2, 0          # j = 0
389
390 add $t1, $t1, $t0    # Cong buoc nhay
391
392 compare_reg:
393 add $t3, $a2, $t1    # t3 tro thanh vi tri tro den dau cua tung
Register
394 lb $t4, 0($t3)
395 beq $t4, 0, not_found
396 beq $t4, 32, check_len_reg    # Neu gap ky tu ' ' => Kiem tra do
dai
397
398 add $t5, $a1, $t2    # Load token
399 lb $t6, 0($t5)
400
401 bne $t4, $t6, compare_token_register    # So sanh 2 ki tu, neu khong
bang nhau thi tinh den Register tiep theo.
402 addi $t1, $t1, 1    # i = i + 1
403 addi $t2, $t2, 1    # j = j + 1
404 j compare_reg
405
406 check_len_reg:
407 bne $t2, $t9, compare_token_register    # Neu do dai khong bang nhau
di den register tiep theo
408
409 end_compare_token_register:
410
411 # >>>>>>>> In thong tin ra man hinh <<<<<<<<<
412 beq $s2, 1, on_token_number_register
413 li $v0, 4
414 la $a0, toanHang_mess
415 syscall
416
417 la $a3, token
418 li $t0, 0
419 print_token_register:
420 beq $t0, $t9, end_print_token_register
421 add $t1, $a3, $t0
422 lb $t2, 0($t1)
423 li $v0, 11
424 add $a0, $t2, $zero
425 syscall
426 addi $t0, $t0, 1
427 j print_token_register
428 end_print_token_register:
429
430 li $v0, 4
431 la $a0, hopLe_mess
432 syscall
433 jr $ra
434
435 on_token_number_register:

```

```

436
437 la $a3, token
438 li $t0, 0
439 print_on_token_register:
440 beq $t0, $t9, end_print_on_token_register
441 add $t1, $a3, $t0
442 lb $t2, 0($t1)
443 li $v0, 11
444 add $a0, $t2, $zero
445 syscall
446 addi $t0, $t0, 1
447 j print_on_token_register
448 end_print_on_token_register:
449
450 li $v0, 11
451 li $a0, 41
452 syscall
453 li $v0, 4
454 la $a0, hopLe_mess
455 syscall
456 jr $ra
457
458 #-----
459 # @check_ident: Kiem tra ident (label) HOAC number
460 # a0: command (vi tri luu command)
461 # a1: ident (vi tri luu ident)
462 # a2: characterGroup | numberGroup
463 # $s7: luu index cua command
464 # $t9: index cua ident
465 #-----
466 check_ident:
467 la $a0, command
468 la $a1, ident
469
470 add $t0, $a0, $s7      # Tro den vi tri cac instruction
471
472 li $t1, 0             # i = 0
473 li $t9, 0             # index cua ident
474
475 read_ident:
476 add $t2, $t0, $t1      # command
477 add $t3, $a1, $t1      # ident
478 lb $t4, 0($t2)
479
480 beq $t4, 40, end_read_ident    # Gap ky tu '('
481 beq $t4, 44, end_read_ident    # Gap ky tu ', '
482 beq $t4, 10, end_read_ident    # Gap ky tu '\n'
483 beq $t4, 0, end_read_ident     # Ket thuc
484
485 addi $t1, $t1, 1
486 beq $t4, 32, read_ident        # Neu gap dau ' ' thi tiep tuc
487

```

```

488 sb $t4, 0($t3)
489 addi $t9, $t9, 1
490 j read_ident
491
492 end_read_ident:
493 add $s7, $s7, $t1      # Cap nhat lai gia tri index
494 beq $t9, 0, not_found  # Khong co label
495
496 #li $v0, 10
497 #syscall
498
499 li $t2, 0              # index cho Ident
500 compare_ident:
501 beq $t2, $t9, end_compare_ident # ket thuc chuoi
502 li $t1, 0              # index cho characterGroup
503
504 add $t3, $a1, $t2
505 lb $t3, 0($t3)         # Tung char trong Ident
506
507 loop_Group:            # Kiem tra tung ky tu Ident co trong Group hay
508     # khong
509     add $t4, $a2, $t1
510     lb $t4, 0($t4)
511     beq $t4, 0, not_found # Khong co -> Khong tim thay
512     beq $t4, $t3, end_loop_Group
513
514     addi $t1, $t1, 1
515     j loop_Group
516
517 end_loop_Group:
518
519 addi $t2, $t2, 1
520
521 j compare_ident
522
523 end_compare_ident:
524
525 beq $s2, 1, on_number_register
526
527 # >>>>>>> In thong tin ra man hinh <<<<<<<<<
528 li $v0, 4
529 la $a0, toanHang_mess
530 syscall
531
532 la $a3, ident
533 li $t0, 0
534 print_ident:
535     beq $t0, $t9, end_print_ident
536     add $t1, $a3, $t0
537     lb $t2, 0($t1)
538     li $v0, 11
539     add $a0, $t2, $zero

```



```

539     syscall
540     addi $t0, $t0, 1
541     j print_ident
542 end_print_ident:
543
544     li $v0, 4
545     la $a0, hopLe_mess
546     syscall
547     jr $ra
548
549 on_number_register:
550     li $v0, 4
551     la $a0, toanHang_mess
552     syscall
553
554     la $a3, ident
555     li $t0, 0
556 print_on_ident:
557     beq $t0, $t9, end_print_on_ident
558     add $t1, $a3, $t0
559     lb $t2, 0($t1)
560     li $v0, 11
561     add $a0, $t2, $zero
562     syscall
563     addi $t0, $t0, 1
564     j print_on_ident
565 end_print_on_ident:
566
567     li $v0, 11
568     li $a0, 40
569     syscall
570     jr $ra
571
572 #-----
573 # @check_number_register: Kiem tra number - ident
574 # a0: command (vi tri luu command)
575 # $s7: luu index cua command
576 # $s2: Luu kich hoat check number register
577 #-----
578
579 check_number_register:
580     # Luu $ra de tro ve
581     addi $sp, $sp, -4
582     sw $ra, 0($sp)
583
584     li $s2, 1          # Bat kich hoat number_register
585
586     # Check number
587     la $a2, numberGroup
588     jal check_ident
589     nop
590

```

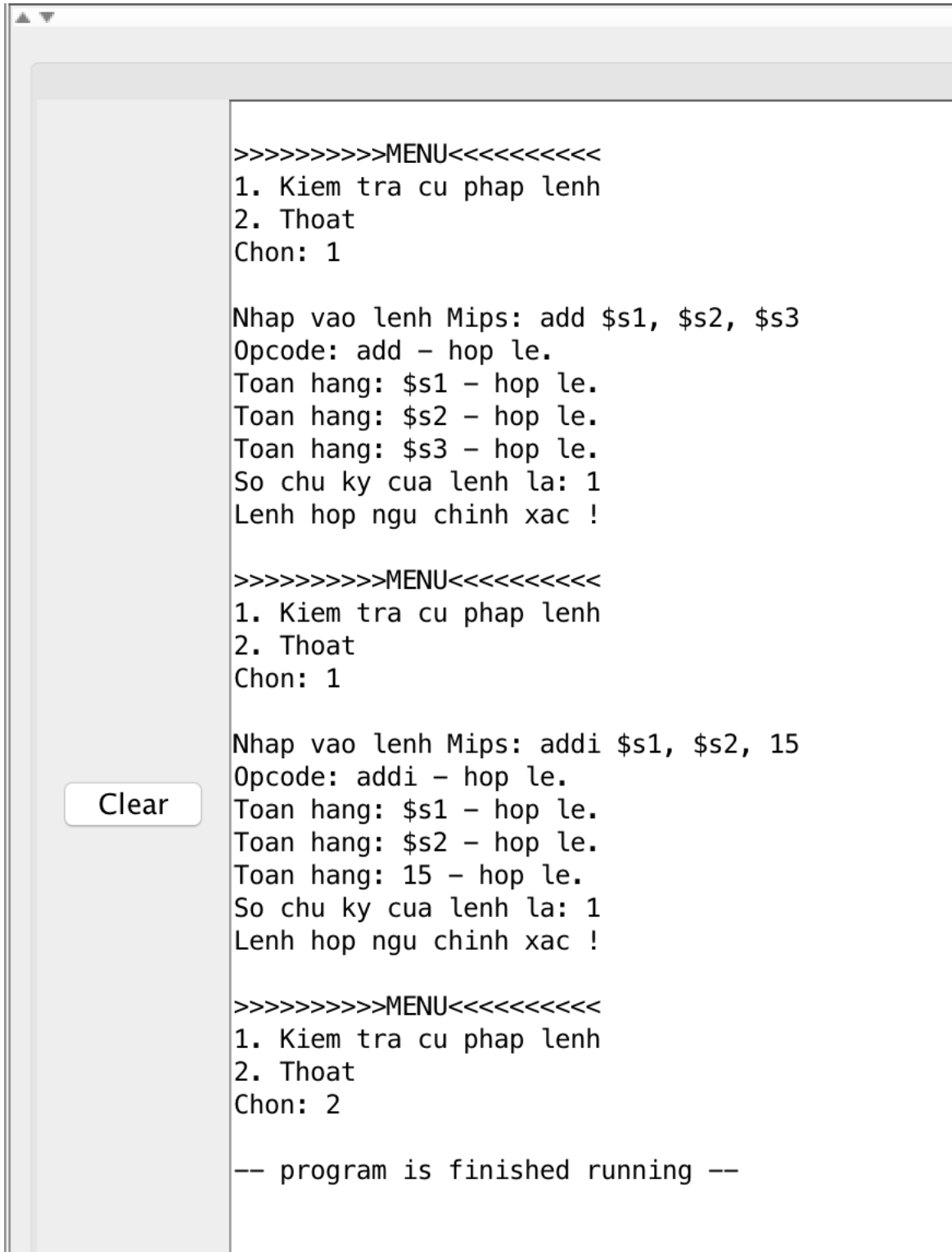
```

591 la $a0, command
592 add $t0, $a0, $s7      # Tro den vi tri cac instruction
593 lb $t0, 0($t0)
594 bne $t0, 40, not_found # Neu ki tu khong phai la dau '('
595 addi $s7, $s7, 1
596
597 # Check register
598 jal check_register
599 nop
600 la $a0, command
601 add $t0, $a0, $s7      # Tro den vi tri cac instruction
602 lb $t0, 0($t0)
603 bne $t0, 41, not_found # Neu ki tu khong phai la dau ')'
604 addi $s7, $s7, 1
605
606 # Tra lai $ra de tro ve
607 lw  $ra, 0($sp)
608 addi $sp, $sp, 4
609 jr $ra
610
611 #-----
612 # @not_found: Khong tim thay khon dang lenh
613 #-----
614 not_found:
615     li $v0, 4
616     la $a0, error_mess
617     syscall
618     j m_menu_start
619
620 #-----
621 # END
622 #-----

```

Project 07 - Chương trình kiểm tra cú pháp lệnh MIPS

2.4 Kết quả chạy chương trình



```
>>>>>>>>MENU<<<<<<<<<
1. Kiem tra cu phap lenh
2. Thoat
Chon: 1

Nhap vao lenh Mips: add $s1, $s2, $s3
Opcode: add - hop le.
Toan hang: $s1 - hop le.
Toan hang: $s2 - hop le.
Toan hang: $s3 - hop le.
So chu ky cua lenh la: 1
Lenh hop ngu chinh xac !

>>>>>>>>MENU<<<<<<<<<
1. Kiem tra cu phap lenh
2. Thoat
Chon: 1

Nhap vao lenh Mips: addi $s1, $s2, 15
Opcode: addi - hop le.
Toan hang: $s1 - hop le.
Toan hang: $s2 - hop le.
Toan hang: 15 - hop le.
So chu ky cua lenh la: 1
Lenh hop ngu chinh xac !

>>>>>>>>MENU<<<<<<<<<
1. Kiem tra cu phap lenh
2. Thoat
Chon: 2

-- program is finished running --
```