CO2008 - KIẾN TRÚC MÁY TÍNH

Khoa Khoa học và kỹ thuật máy tính
Đại học Bách Khoa - ĐHQG Tp.HCM

09/2024

Bài thực hành 3

CHƯƠNG 2 KIẾN TRÚC TẬP LỆNH MIPS: Các lệnh điều khiển

Mục tiêu

- Chuyển từ ngôn ngữ cấp cao (C) sang hợp ngữ MIPS.
- Sử dụng lệnh điều khiển (nhảy, rẽ nhánh) để điều khiển luồng thi hành chương trình.

Yêu cầu

- Xem lại hoạt động của các lệnh set, branch, jump, load, store.

Tham khảo nhanh các lệnh

Cú pháp	Hành động	Diễn giải
`slt Rd,Rs,Rt`	`Rd=(Rs<Rt)?1:0`	`Rd=1(có dấu) nếu Rs<Rt;ngược lại Rd=0`
`sltu Rd,Rs,Rt`	`Rd=(Rs<Rt)?1:0`	`Rd=1(không dấu) nếu Rs<Rt;ngược lại Rd=0`
Lệnh nhảy, rẽ nhánh		
`beq Rs,Rt,label`	`if Rs=Rt then branch`	`Rẽ nhánh đến label nếu Rs=Rt`
`bne Rs,Rt,label`	`if Rs≠Rt then branch`	`Rẽ nhánh đến label nếu Rs≠Rt`
`bltz Rs,label`	`if Rs<0 then branch`	`Rẽ nhánh đến label nếu Rs<0`
`blez Rs,label`	`if Rs<=0 then branch`	`Rẽ nhánh đến label nếu Rs<=0`
`bgtz Rs,label`	`if Rs>0 then branch`	`Rẽ nhánh đến label nếu Rs>0`
`bgez Rs,label`	`if Rs>=0 then branch`	`Rẽ nhánh đến label nếu Rs>=0`
`j label`	`jump to label`	`Nhảy không điều kiện đến label`
`jr Rs`	`PC←Rs`	`Nhảy không điều kiện đến địa chỉ chứa trong thanh ghi Rs`
Lệnh gọi hàm, trở về		
`jal label`	`$ra←return address; jump to label`	`Gọi hàm label: lưu địa chỉ lệnh tiếp theo vào $ra, sau đó nhảy đến label`
`jalr Rs`	`$ra←return address; PC←Rs`	`Gọi hàm gián tiếp qua thanh ghi Rs (cần nạp địa chỉ hàm vào Rs trước): lưu địa chỉ lệnh tiếp theo vào $ra, sau đó chuyển Rs vào PC.`
`jr $ra`	`PC←$ra`	`Trở về từ jal`

Bài tập và thực hành

Lập trình có cấu trúc (tham khảo các lưu đồ bên dưới).

Sinh viên chuyển các cấu trúc sau của ngôn ngữ C qua hợp ngữ MIPS. Tham khảo lưu đồ các cấu trúc ở cuối bài. Viết thành chương trình hoàn chỉnh, định nghĩa các biến cần thiết, nhập dữ liệu ban đầu cho các biến, thực hiện cấu trúc và in kết quả ra màn hình.

Bài 1. Phát biểu IF-ELSE (1)

```
if(a%2==0)    {print string "Computer Science and Engineering, HCMUT."}
else          {print string "Computer Architecture 2024."}
```

Bài 2. Phát biểu IF-ELSE (2)

```
if (a>=-3 && a<=4) a=b-c;
else a=b+c;
```

Bài 3. Phát biểu SWITCH-CASE.

Hiện thực phát biểu switch-case bên dưới bằng hợp ngữ. Cho biết b=100, c=2. Giá trị input nhập từ người dùng. Xuất ra giá trị của a.

```
switch (input)
{   case 1: a = b + c; break;
    case 2: a = b - c; break;
    case 3: a = b * c; break;
    case 4: a = b / c; break;
    default: a = 0;
}
```

Bài 4. Vòng lặp FOR.

Tính chuỗi Fibonacci bằng vòng lặp và lưu vào dãy f[] có 20 phần tử. Nhập vào số n (0÷19), xuất phần tử f[n] ra màn hình. Giải thuật tính dãy Fibonacci như sau:

```
f[0]=0; f[1]=1;
for(i=2;i<n;i++){ f[i]=f[i-1]+f[i-2];}
```

Dùng bảng sau để kiểm tra lại kết quả chạy chương trình:

F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9
0	1	1	2	3	5	8	13	21	34
F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}	F_{16}	F_{17}	F_{18}	F_{19}
55	89	144	233	377	610	987	1597	2584	4181

Bài 5. Vòng lặp WHILE.

Xác định vị trí chữ 'u' đầu tiên trong chuỗi "Computer Architecture CSE-HCMUT".

```
i=0;
while(charArray[i]!='u' && charArray[i]!='\0') i++;
```

Dùng if kiểm tra lại điều kiện để xuất ra màn hình vị trí tìm thấy ký tự hoặc thông báo không tìm thấy.

Làm thêm

1. ENDIANESS.

   Cho mảng số nguyên bên dưới:

```
.data
intArray: .word 0xCA002019,0xC0002009
.text
          la $a0,intArray
          lb $t0,0($a0)
          lb $t1,1($a0)
          lb $t2,2($a0)
          lb $t3,3($a0)
          lbu $t4,0($a0)
          lbu $t5,1($a0)
          lbu $t6,2($a0)
          lbu $t7,3($a0)
```

   a) Giả sử MIPS được thiết kế theo kiểu BIG ENDIAN, xác định giá trị các ô nhớ (theo byte) của mảng trên.

   b) Giả sử MIPS được thiết kế theo kiểu LITTLE ENDIAN, xác định giá trị các ô nhớ (theo byte) của mảng trên.

   c) Xác định giá trị các thanh ghi $t của đoạn code bên dưới, giả sử MIPS được thiết kế theo kiểu BIG ENDIAN.

   d) Xác định giá trị các thanh ghi $t của đoạn code bên dưới, giả sử MIPS được thiết kế theo kiểu LITTLE ENDIAN.

2. Memory alignment.

   Cho đoạn code MIPS bên dưới:

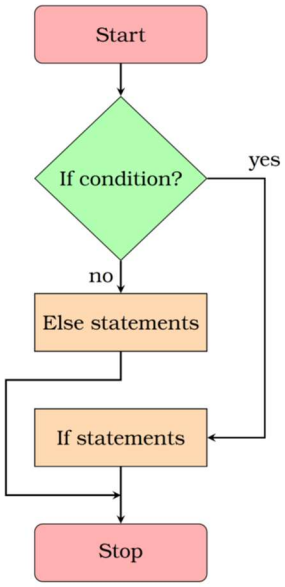
```
.data
int_1:    .word 0xCA002018
char_1:   .byte 0xFF
int_2:    .word 2018
char_2:   .byte 0xCA 0xFE 0xED
```

```
.text
        la $a0,int_1
        lw $t0,0($a0)
        lw $t1,1($a0)
        lh $t2,2($a0)
        lh $t3,3($a0)
        lb $t4,0($a0)
        lb $t5,1($a0)
```

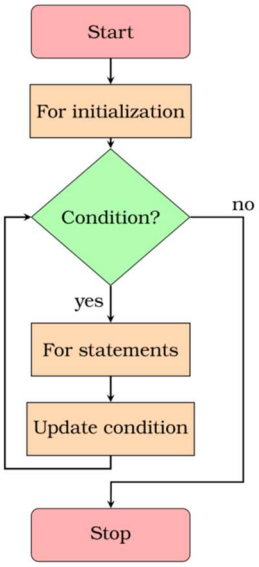
a) Xác định nội dung của vùng nhớ dữ liệu và xác định các lệnh sẽ gây ra lỗi khi thực thi, giải thích. Biết MIPS chuẩn được thiết kế theo kiểu BIG ENDIAN.

b) Sắp xếp lại dữ liệu sao cho bộ nhớ tối ưu hơn (trong kiến trúc 32 bit).

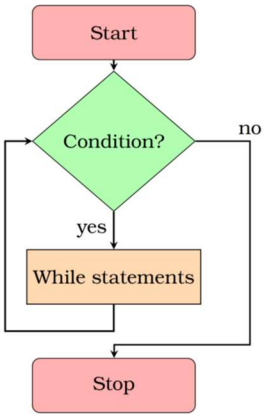
Lưu đồ các cấu trúc if-else, for, while, do-while



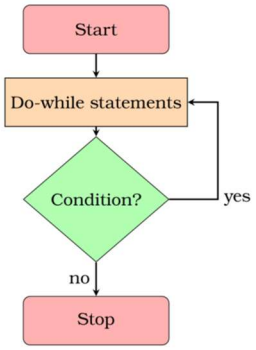
Hình. 1: If-else statement



Hình. 2: For statements



Hình. 3: While statement



Hình. 4: Do-while statement