

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA



Computer Architecture (CO2008)

Bài báo cáo Chapter 3.1 (Lab 5)

Họ và tên	Nguyễn Phúc Tiến
MSSV	2014725
Lớp	L03

Ngày 29 tháng 4 năm 2022

Mục lục

1	Số thực IEEE 754	2
1.1	Xác định giá trị số thực được chứa trong thanh ghi có nội dung 0xCA202000.	2
1.2	Xác định nội dung thanh ghi mà giá trị số thực của nó là 36.15625.	2
1.3	Có thể biểu diễn chính xác giá trị 20.2 ở dạng IEEE không? giải thích. Khoảng cách giữa 2 số thực liên tiếp (biểu diễn được bằng IEEE 754) có bằng nhau không? Giải thích. . . .	2
2	Tính chu vi và diện tích hình tròn với bán kính là số thực	3
2.1	Đề bài	3
2.2	Hiện thực	3
3	Tìm min, max của mảng số thực	4

1 Số thực IEEE 754

1.1 Xác định giá trị số thực được chứa trong thanh ghi có nội dung 0xCA202000.

Đầu tiên ta convert mã Hex sang Binary:

$$0xCA202000 = 11001010001000000010000000000000$$

- Bit đầu tiên là bit dấu: 1 tương ứng với bit âm (1^{-1}).
- 8 bits Exponent: 10010100_2 tương đương với $148_{10} \Rightarrow e = 148 - 127 = 21$.
- 23 bits mantissa 01000000010000000000000 : $m = 2^{-2} + 2^{-10} = 0.2509765625$.
 \Rightarrow Giá trị số thực $= -1.(1 + m).2^e = -1.2509765625 \times 2^{21} = -2623488$

1.2 Xác định nội dung thanh ghi mà giá trị số thực của nó là 36.15625.

- Là số dương \Rightarrow bit dấu $s = 0$.
- Phần nguyên $= 36_{10} = 100100_2$
- Phần thập phân $= 0.15625_{10} = 0.00101_2$
 $\Rightarrow 36.15625_{10} = 100100.00101_2$
- Sau đó dịch dấu chấm động sang bên trái ta được: $1.0010000101 \times 2^5 \Rightarrow E = 5$.
 $\Rightarrow e = 127 + E = 132_{10} = 10000100_2$

\Rightarrow Số thực 36.15625 vừa nhập chuyển về dạng chuẩn IEEE 754/85. Với màu **xanh dương** thể hiện bit dấu, màu **xanh lá** thể hiện 8 bits exponent và màu **đỏ** thể hiện 23 bits mantissa.

$$01000010000100001010000000000000 = 0x4210A000$$

1.3 Có thể biểu diễn chính xác giá trị 20.2 ở dạng IEEE không? giải thích. Khoảng cách giữa 2 số thực liên tiếp (biểu diễn được bằng IEEE 754) có bằng nhau không? Giải thích.

- Không thể biểu diễn chính xác giá trị 20.2 ở dạng IEEE vì phần thập phân $= 0.2$ khi chuyển sang binary sẽ là $0.(0011)$, là số vô hạn tuần hoàn, mà bit mantissa chỉ lấy 23 bits, nên còn các bit phía sau không được lấy. Do đó không thể biểu diễn chính xác giá trị 20.2 ở dạng IEEE.
- Biểu diễn số thực ở dạng IEEE là:

$$2^e \cdot (1 + n \cdot 2^{-23})$$

Trong đó $n \in \{0, 1, \dots, 2^{23} - 1\}$. Cho nên trừ 2 số cho nhau thì khoảng cách của 2 số thực liên tiếp luôn bằng nhau và bằng 2^{e-23} . Tuy nhiên với điều kiện là trong khoảng 2^{23} bits (độ chính xác đơn). Khi ra khỏi số bits đó sẽ làm tròn lên số gần nhất (độ chính xác kép là 2^{53} bits).

2 Tính chu vi và diện tích hình tròn với bán kính là số thực

2.1 Đề bài

Viết chương trình nhập vào bán kính đường tròn (số thực). Xuất ra chu vi và diện tích của hình tròn đó (chú ý trường hợp số âm và zero).

2.2 Hiện thực

- Đầu tiên ta sẽ khai báo những string để giao tiếp với người dùng và những giá trị số thực như số pi, zero,...

```
1 .data
2 cMessage: .ascii "\nChu vi cua hinh tron la: "
3 sMessage: .ascii "\nDien tich cua hinh tron la: "
4 message: .ascii "Nhap vao ban kinh hinh tron : "
5 Error: .ascii "\nVui long nhap ban kinh la mot so khong am"
6 pi: .float 3.14
7 TwoAsFloat: .float 2.0
8 ZeroAsFloat: .float 0.0
```

- Sau đó trong phần .data ta sẽ tiến hành đọc bán kính R từ người dùng.

```
1 .text
2 #Display the message
3 li $v0, 4
4 la $a0, message
5 syscall
6
7 #Enter user's floating point radius
8 li $v0, 6
9 syscall
```

- Tiếp theo ta sẽ gán những giá trị như là π , zero, 2 ở dạng số thực để thực hiện các phép toán với số thực. Thanh ghi \$f1 sẽ chứa giá trị π , \$f2 là giá trị 2 dạng float và \$f10 chứa giá trị zero dạng float.

```
1 #Store pi, zero and two value into register
2 lwc1 $f1, pi
3 lwc1 $f2, TwoAsFloat
4 lwc1 $f10, ZeroAsFloat
```

- Vì bán kính hình tròn là một số không âm do đó ta sẽ kiểm tra điều kiện ($R < 0$) và sẽ xuất ra lỗi nếu người dùng nhập giá trị âm. Ta sử dụng lệnh c.lt.s để so sánh bé hơn, nếu \$f0 mà bé hơn \$f10 thì condition flag 1 sẽ bằng 1, ngược lại sẽ bằng 0. Sau đó ta sử dụng lệnh bc1t để kiểm tra flag = 1 hay 0. Nếu bằng 1 thì sẽ tiến hành rẽ nhánh.

```
1 #check if user's radius is negative
2 c.lt.s $f0, $f10 #flag = 1 if radius < 0
3 bc1t exit #branch if true
4
5 exit:
6 #Display error message
7 li $v0, 4
8 la $a0, Error
9 syscall
10
```

- Sau khi kiểm tra điều kiện R thì ta sẽ tính toán chu vi và diện tích. Với chu vi thì giá trị sẽ được gán vào trong thanh ghi \$f4 còn diện tích thì gán vào trong thanh ghi \$f5.

```
1 #Calculate circumference
2 mul.s $f4, $f1, $f0 # pi * R
3 mul.s $f4, $f4, $f2 # (pi * R) * 2
4
5 #Calculate Area
6 mul.s $f5, $f1, $f0 # pi * R
7 mul.s $f5, $f5, $f0 # (pi * R) * R
```

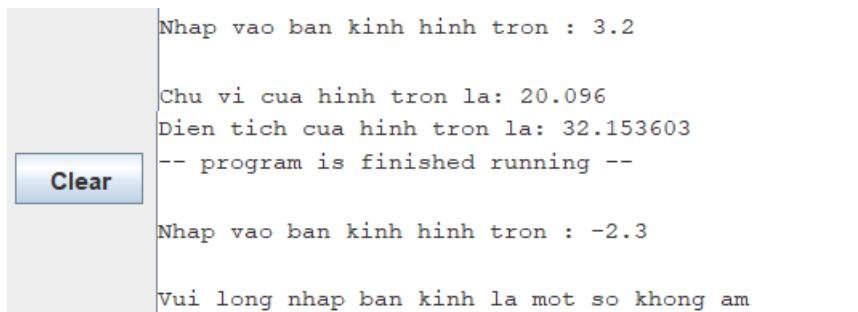
- Cuối cùng là ta sẽ print ra message cho chu vi và diện tích và in giá trị ra console. Với floating point number thì arguments sẽ là thanh ghi `$f12`.

```

1  #Display message and print result
2  li      $v0, 4
3  la      $a0, cMessage
4  syscall
5  #Print Circumference of circle
6  li      $v0, 2
7  add.s   $f12, $f4, $f10
8  syscall
9
10 li      $v0, 4
11 la      $a0, sMessage
12 syscall
13 #print Area of circle
14 li      $v0, 2
15 add.s   $f12, $f5, $f10
16 syscall
17
18 li      $v0, 10    #exit()
19 syscall

```

Kết quả:



```

Nhap vao ban kinh hinh tron : 3.2

Chu vi cua hinh tron la: 20.096
Dien tích của hình tron la: 32.153603
-- program is finished running --

Nhap vao ban kinh hinh tron : -2.3

Vui long nhap ban kinh la mot so khong am

```

3 Tìm min, max của mảng số thực

Cho mảng số thực 20 phần tử, xác định giá trị lớn nhất, nhỏ nhất của mảng.

- Đầu tiên trong phần `.data` ta cũng tạo các string, mảng số thực tên là `fArray` chứa 20 phần tử và zero dạng float.

```

1  .data
2  maxMessage: .asciiz "\nGia tri lon nhat la: "
3  minMessage: .asciiz "\nGia tri nho nhat la: "
4  .align 4
5  fArray: .space 80
6  ZeroAsFloat: .float 0.0
7  space: .asciiz " "
8

```

- Tiếp theo ta sẽ tạo một mảng `fArray` chứa số thực ngẫu nhiên từ 0 - 1 bằng cách cách load giá trị 43 cho thanh ghi `$v0` và dùng lệnh `s.s` để store float vào trong `fArray`.

```

1  generate_fArray:
2  li      $a0, 0
3  li      $v0, 43    #service 43 is generate random float and store in $f0
4  syscall
5
6  beq     $t0, 80, exit_generate # if index = 80 then exit
7  s.s     $f0, fArray($t0)      # store f0 into fArray at index i
8  addi    $t0, $t0, 4          # index++
9  j       generate_fArray
10 exit_generate:
11

```

- Sau đó print fArray ra để xem những phần tử có trong mảng.

```

1 # Print fArray to see these random numbers
2 li      $t0, 0      # reset index i
3 print_fArray:
4     beq      $t0, 80, exit_print # if index = 80 then exit
5     l.s      $f2, fArray($t0)    # f2 contains the value of fArray[i]
6     addi     $t0, $t0, 4          # index++
7     # Print fArray[i]
8     li      $v0, 2              # service 2 is print float
9     add.s    $f12, $f2, $f10    # $f12 is argument
10    syscall
11
12    li      $v0, 4              # print space: " "
13    la      $a0, space
14    syscall
15    j        print_fArray
16 exit_print:
17

```

- Tìm **max**: Đầu tiên ta gán $\text{max} = \text{fArray}[0]$ bằng lệnh `lwcl` (Load word coprocessor 1) và được gán trong thanh ghi `$f1`.

```

1      li      $t0, 0      # reset index i = 0
2      lwcl    $f1, fArray($zero) # $f1 = max = fArray[0]
3      # find max
4      jal     max
5

```

Chương trình jump and link đến label `max`:. Tại đây ta thực hiện việc tìm max. Sử dụng lệnh `l.s` để load giá trị float của `fArray[i]` và được gán vào trong thanh ghi `$f5`. Sau đó sử dụng so sánh max với `$f5` bằng lệnh `c.lt.s`. Nếu bé hơn thì condition flag 1 sẽ bằng 1, ngược lại bằng 0. Nếu $\text{max} < \text{fArray}[i]$ thì ta jump không điều kiện về label `max` và $\text{index}++$, ngược lại $\text{max} = \text{fArray}[i]$. Cuối cùng dùng `jr $ra` để jump về địa chỉ mà thanh ghi `$ra` đã đánh dấu.

```

1 max:
2     addi     $t0, $t0, 4          # index++
3     beq      $t0, 80, exit_find_max # condition to exit find_max loop
4     l.s      $f5, fArray($t0)    # load fArray[i] and store in $f5
5     c.lt.s   $f1, $f5            # max < fArray[i]? 1 : 0
6     bc1f     max                 # if 0 then jump to max label
7     lwcl    $f1, fArray($t0)    # else max = fArray[i]
8     j        max
9 exit_find_max:
10    jr       $ra
11

```

- Tìm **min**: Cách làm tương tự việc tìm max, ta gán giá trị min vào thanh ghi `$f2`.

```

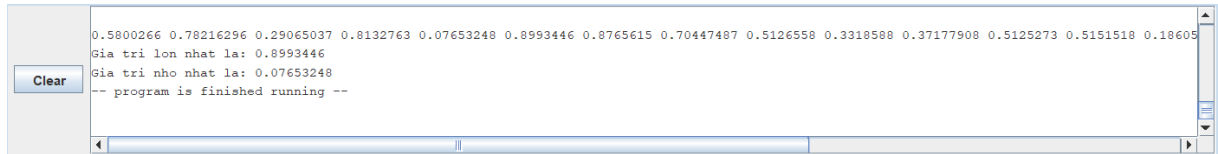
1      li      $t0, 0      # reset index value
2      lwcl    $f2, fArray($zero) # min = fArray[0]
3      jal     min
4
5 #find min of fArray
6 min:
7     addi     $t0, $t0, 4          # index++
8     beq      $t0, 80, exit_find_min # condition to exit find_min loop
9     l.s      $f5, fArray($t0)    # load fArray[i] and store in $f5
10    c.lt.s   $f2, $f5            # min < fArray[i]? 1 : 0
11    bc1t     min                 # if 1 then jump to min label
12    lwcl    $f2, fArray($t0)    # else min = fArray[i]
13    j        min
14 exit_find_min:
15    jr       $ra
16

```

- Sau khi jump về địa chỉ mà thanh ghi `$ra` đã đánh dấu là sau `jal min`. Ta tiến hành in ra các ra kết quả max được chứa trong thanh ghi `$f1` và min được chứa trong thanh ghi `$f2`.

```
1      # Display message
2      li      $v0, 4
3      la      $a0, maxMessage
4      syscall
5      # Print max
6      li      $v0, 2
7      add.s   $f12, $f1, $f10
8      syscall
9
10     # Display message
11     li      $v0, 4
12     la      $a0, minMessage
13     syscall
14     # Print max
15     li      $v0, 2
16     add.s   $f12, $f2, $f10
17     syscall
18
19
20     li      $v0, 10  #exit()
21     syscall
22
```

Kết quả:



```
1      0.5800266 0.78216296 0.29065037 0.8132763 0.07653248 0.8993446 0.8765615
2      0.70447487 0.5126558 0.3318588 0.37177908 0.5125273 0.5151518 0.1860593
3      0.5626158 0.4097696 0.4372304 0.16439235 0.32194585 0.51786464
4      Gia tri lon nhat la: 0.8993446
5      Gia tri nho nhat la: 0.07653248
6      -- program is finished running --
```