Mini-project KTMT

Bài 3 + Bài 10

Đào Minh Nhật – 20215107

Nguyễn Trọng Nghĩa - 20215101

Bài 3:

Code:

```
#Mini_project Bai 3
.data
        prompt: .asciiz "Nhap so: "
        result: .asciiz "Ket qua: "

        zero: .asciiz "zero "
        one: .asciiz "one "
        two: .asciiz "two "
        three: .asciiz "three "
        four: .asciiz "four "
        five: .asciiz "five "
        six: .asciiz "six "
        seven: .asciiz "seven "
        eight: .asciiz "eight "
        nine: .asciiz "nine "
        ten: .asciiz "ten "
        eleven: .asciiz "eleven "
        twelve: .asciiz "twelve "
        thirteen: .asciiz "thirteen "
        fourteen: .asciiz "fourteen "
        fifteen: .asciiz "fifteen "
        sixteen: .asciiz "sixteen "
        seventeen: .asciiz "seventeen "
        eighteen: .asciiz "eighteen "
        nineteen: .asciiz "nineteen "
```

```
        digits: .word zero, one, two, three, four, five, six, seven,
                eight, nine, ten, eleven, twelve, thirteen, fourteen,
                 fifteen, sixteen, seventeen, eighteen, nineteen

        twenty: .asciiz "twenty "
        thirty: .asciiz "thirty "
        forty: .asciiz "forty "
        fifty: .asciiz "fifty "
        sixty: .asciiz "sixty "
        seventy: .asciiz "seventy "
        eighty: .asciiz "eighty "
        ninety: .asciiz "ninety "
        ge20: .word twenty, thirty, forty, fifty, sixty, seventy, eighty, ninety

        hundred: .asciiz "hundred "
        thousand: .asciiz "thousand "
        million: .asciiz "million "
        andd: .asciiz "and "
#-------------------------------------------------------------
# @brief Bai 3 su dung 2 mang digits va mang ge20 (greater or equal 20)
# @param mang digits chua dia chi tro toi xau cac so tu 0-19
# @param mang ge20 chua dia chi tro toi xau cac so hang chuc
# @param cac bien hundred, thousand, million, andd la tu noi
# @note dung mang se tien de truy nhap xau hon
#-------------------------------------------------------------
```

```
.text
main:
#-------------------------------------------------------------
# @brief chuong trinh nhap va in ra man hinh nhu de bai
# @param so nguyen nhap vao duoc luu vao $s0
# @param $ra luu lai dia chi cau lenh va nhay den chuong trinh in
#-------------------------------------------------------------
        li $v0, 4
        la $a0, prompt
        syscall

        li $v0, 5
        syscall
        move $s0, $v0

        li $v0, 4
        la $a0, result
        syscall

        move $a0, $s0
        jal convert_to_text
#-------------------------------------------------------------
# @brief Ket thuc chuong trinh
#-------------------------------------------------------------
        li $v0, 10
```

```
76          syscall
77  End_main:
78  #-------------------------------------------------------------
79  # @brief Chia so thanh hang trieu, hang nghin, va nho hon 1000
80  # @param Khoi tao stack va luu $ra vao stack
81  # @param Cac chuong trinh skip_million, skip_thousand kiem tra cac so >=1000000 va >=1000
82  # @param Nhay den chuong trinh print_below_thousand de in bo ba so
83  # @param Sau khi in xong bo ba so se in them tu noi
84  # @note cap phat bo nho dong, luu dia chi tra ve cho bien dem
85  # @note khoi phuc lai bo nho dong sau khi da thuc hien xong
86  #-------------------------------------------------------------
87  convert_to_text:
88          addi $sp, $sp, -4
89          sw $ra, 0($sp)
90
91          li $t1, 1000000
92          div $a0, $t1
93          mfhi $t0
94          mflo $t2
95          beqz $t2, skip_million
96
97          addi $sp, $sp, -4
98          sw $t0, 0($sp)
99
100         move $a0, $t2
```

```
101         jal print_below_thousand
102         la $a0, million
103         li $v0, 4
104         syscall
105
106         lw $t0, 0($sp)
107         addi $sp, $sp, 4
108
109 skip_million:
110         move $a0, $t0
111
112         li $t1, 1000
113         div $a0, $t1
114         mfhi $t0
115         mflo $t2
116         beqz $t2, skip_thousand
117
118         addi $sp, $sp, -4
119         sw $t0, 0($sp)
120
121         move $a0, $t2
122         jal print_below_thousand
123         la $a0, thousand
124         li $v0, 4
125         syscall
```

```
126
127         lw $t0, 0($sp)
128         addi $sp, $sp, 4
129
130 skip_thousand:
131         move $a0, $t0
132         jal print_below_thousand
133         lw $ra, 0($sp)
134         addi $sp, $sp, 4
135
136 End_convert_to_text:
137         jr $ra
138 #-------------------------------------------------------------
139 # @brief Chuong trinh in bo ba so
140 # @brief cap phhat bo nho dong vao stack
141 # @param chia $a0 cho 100, lay phan du luu vao $t0, phan thuong luu vao $t2
142 # @param Neu $t2 > 0 thi chuyen den print_digits de in hang tram
143 # @param Neu $t0 != 0 thi nhay den chuong trinh in hang chuc va hang don vi
144 # @return $a0 luu chu so hang tram dong thoi luu hai chu so hang chuc va hang don vi
145 #-------------------------------------------------------------
146 print_below_thousand:
147         addi $sp, $sp, -4
148         sw $ra, 0($sp)
149         li $t1, 100
150         div $a0, $t1
```

```
151         mfhi $t0
152         mflo $t2
153         beqz $t2, skip_hundred
154
155         move $a0, $t2
156         jal print_digits
157         la $a0, hundred
158         li $v0, 4
159         syscall
160
161         beqz $t0, skip_hundred
162         la $a0, andd
163         li $v0, 4
164         syscall
165
166 skip_hundred: nop
167         move $a0, $t0
168         jal print_digits
169         lw $ra, 0($sp)
170         addi $sp, $sp, 4
171         jr $ra
172 End_print_below_thousand:
173 #-------------------------------------------------------------
174 # @brief Chuong trinh in hang chuc, hang don vi va chu so
175 # @param Cap phat bo nho dong vao stack
```

```
# @param Neu $a0 < 20 thi xau can tim o mang digits
# @param Neu $a0 >=20 thi xau can tim o mang ge20
# @param $t3 = 4*$a0 de tro den o nho can tim trong mang digits
# @param $t3 = 4*($a0-2) de tro den o nho can tim trong mang ge20
# @param $a1 load dia chi o nho ma $t3 tro den, chua con tro tro den ky tu dau cua xau
# @param $a0 load dia chi xau va in ra man hinh
# @param Cuoi cung se khoi phuc lai bo nho dem
#-------------------------------------------------------------
print_digits:
        addi $sp, $sp, -4
        sw $ra, 0($sp)

        bge $a0, 20 print_dozen
        li $v0, 4
        sll $t3, $a0, 2
        la $a1, digits($t3)
        lw $a0, ($a1)
        syscall

end:
        lw $ra, 0($sp)
        addi $sp, $sp, 4
        jr $ra

End_print_digits:
```

```
#-------------------------------------------------------------
# @brief Chuong trinh in so hang chuc
# @param Lay $a0 chia 10 de lay phan du $t0 va thuong so $t2
# @param thuong so luu vao $t2 de lay vi tri o nho trong mang ge20
# @note Neu co phan du thi se co hang don vi, vi vay se nhay den ham print_digits
#-------------------------------------------------------------
print_dozen:
        addi $sp, $sp, -4
        sw $ra, 0($sp)

        li $t1, 10
        div $a0, $t1
        mfhi $t0
        mflo $t2
        blt $t2, 1 print_digits

        li $v0, 4
        addi $t2, $t2, -2
        sll $t3, $t2, 2
        la $a1, ge20($t3)
        lw $a0, ($a1)
        syscall

        move $a0, $t0
        beqz $a0, End
```

```
        jal print_digits

End:
        lw $ra, 0($sp)
        addi $sp, $sp, 4
        jr $ra

```

Giải thích:

Khai báo cách đếm số từ 0 đến 19 và lưu trong mảng digits

Khai báo cách đếm của 20, 30, … 90 và lưu trong mảng ge20

Cách làm: tách lần lượt 3 số và đọc.

+ Kiểm tra phần tử triệu:

Lấy $a0 \ 1000000

Nếu thương = 0 thì chuyển sang kiểm tra phần tử nghìn

Ngược lại thương != 0 thì gọi tới hàm "print_below_thousand " để đọc số 3 số hàng triệu sau đó in thêm đơn vị triệu.

+Kiểm tra phần tử nghìn : tương tự phần tử triệu, lấy $a0 \ 1000

+ Hàm "print_below_thousand ": in cách đọc của phần tử < 1000.

Lấy $a0 \ 100 để đọc hàng trăm nếu có. Sau đó đọc hàng chục và hàng đơn vị.

+ Hàm "print_digits": In hàng chục, hàng đơn vị

Nếu $a0 < 20 thì xâu cần tìm ở mảng digits

Ngược lại $a0 >= 20 thì thực hiện hàm "print_dozen", lấy xâu cần tìm ở mảng ge20

+ Hàm "print_dozen": in cách đọc của các số hàng chục từ 20 đến 90.

Nếu có hàng đơn vị thì quay lại hàm print_digits để đọc số hàng đơn vị.

Kết quả:

```
Nhap so: 1432
Ket qua: one thousand four hundred and thirty two
-- program is finished running --
```

Bài 10:

Ở project 10 có 3 functions chính cần phải làm là: hàm mũ 2^i , i^2 và functions chuyển từ số thập phân sang thập lục phân với mà dạng thập lục phân phải ở dạng đơn giản nhất.

Với hàm mũ 2^i, ta khởi tạo một thanh ghi có giá trị ban đầu là 1, một thanh ghi lưu lại giá trị i. Sau đó, ta sử dụng vòng lặp i vòng để dịch trái thanh ghi có giá trị ban đầu là 1, đồng thời cũng giảm giá trị i sau mỗi vòng lặp đến khi i = 0 thì dừng.

Trong không gian 32-bit, ta có thể dịch tối đa 31 lần để tránh hiện tượng tràn. Vì vậy số i nhập vào có giá trị trong khoảng [0; 31] để chương trình được thực hiện. Với i<0 thì chưa nghĩ ra giải pháp.

Với hàm mũ i^2, ta không thể dùng các phép toán logic để tính toán mà phải dùng phép nhân số học (mul) để thực hiện tính toán.

Vì $i^2 = (-i)^2 = |i|^2$ nên nếu người dùng nhập vào một số nguyên âm thì có thể dùng một hàm tính trị tuyệt đối bằng cách lấy số bù 2.

Trong không gian 32-bit, số dương lớn nhất có thể có là $2^{31} - 1$. Vì vậy nên số i sẽ nằm trong khoảng $(-2^{16}, 2^{16})$ tức (-65536, 65536). Vì có số dương lớn nên khi nhân với nhau phải dùng lệnh mulou.

Với hàm chuyển đổi số thập phân sang số thập lục phân ở dạng tối giản, ta sử dụng vòng lặp do-while lưu kết quả vào mảng 8 ô nhớ lưu trữ các ký tự sau khi chuyển đổi và dùng vòng lặp in ra màn hình từng ô nhớ để hiển thị kết quả.

Ta khởi tạo một thanh ghi có giá trị từ 0-32 để chỉ vị trí ô nhớ trong mảng kết quả.

Ta sử dụng một thanh ghi có giá trị 0xF rồi sử dụng phép toán and với i để lấy nội dung 4 bit cuối, lưu vào một thanh ghi khác. Thanh ghi này sẽ cập nhật giá trị của mình sao cho đúng với ký tự trên bảng mã ASCII. Nếu thanh ghi có giá trị từ 48 đến 57 thì là số từ ‘0’-’9’, nếu không thì là chữ cái, cộng thêm 7 để có giá trị từ 65 đến 70. Sau đó

lưu giá trị vào mảng kết quả, tăng giá trị thanh ghi trỏ đến vị trí ô nhớ trong mảng thêm 4 để sang ô nhớ tiếp theo đồng thời cũng dịch phải thanh nhớ chứa giá trị i.

Đến khi i = 0 hoặc thanh ghi trỏ đến vị trí ô nhớ đạt giá trị 32 thì dừng vòng lặp do-while. Nếu người dùng nhập i=0 thì chương trình vẫn hiện ra '0x0'.

Vòng lặp in kết quả sau đó sẽ lấy kết quả từ thanh ghi trỏ đến vị trí ô nhớ trong mảng, từ đó truy ngược lại đến ô nhớ đầu tiên, mỗi lần truy ngược lại sẽ in ra màn hình ký tự trong bảng ASCII.

Source code:

```
.data
        msg1: .asciiz "Enter an integer: "
        msg2: .asciiz "i\t\t\tpower(2,i)\t\tsquare(i)\t\tHexadecimal(i)\n"
        msg3: .asciiz "\t\t\t"
        msg4: .asciiz "0x"
        msg5: .asciiz "Overflow Result"
        hex: .space 8
.text
        li      $v0, 4
        la      $a0, msg1
        syscall
#---------------------------------------------------------------
# @brief Nhap vao mot so nguyen 32-bit tu ban phim
# @param $v0 goi service tu he thong de nhap so nguyen
# @param $v0 chua so da nhap vao
# @return $a1 luu so nguyen da duoc nhap
#---------------------------------------------------------------
        li      $v0, 5
        syscall
        move $a1, $v0
```

```
    move $a1, $v0
#---------------------------------------------------------------
# @brief Chuong trinh in bang theo de bai
# @param $v0 goi service tu he thong de in bang
# @param $a0 luu dia chi cac messages tu phan data
# @return in ra bang
#---------------------------------------------------------------
print_head:
    li  $v0, 4
    la  $a0, msg2
    syscall
print_result:
    li  $v0, 1
    move $a0, $a1
    syscall

    li  $v0, 4
    la  $a0, msg3
    syscall
#---------------------------------------------------------------
# @brief Neu ket qua bi tran thi khong in ra ket qua
```

```
# @brief Nhay den function pow, lay ket qua va in ra
# @param De in duoc ket so nguyen duong lon thi phai in so nguyen duong khong dau
# @param Ket qua cua function pow duoc luu o $t1
# @return In ra man hinh ket qua o $a0 neu khong bi tran
# @note Trong he so hoc 32 bit chi co the hien thi gia tri cao nhat la pow(2, 31)
# @note Vi pow(2, 1) = 1 nen chi duoc dich trai toi da 31 lan, i>31 se xay ra hien tuong tran so
#-------------------------------------------------------------
print_pow:
    bgt $a1, 31, pow_over
    bltz $a1, pow_over
    jal pow_func
    li  $v0, 36
    move $a0, $t1
    syscall
    j   exit_pow
pow_over:
    la  $a0, msg5
    syscall

exit_pow:
    li  $v0, 4
    la  $a0, msg3
    syscall
#-------------------------------------------------------------
# @brief Neu ket qua bi tran thi khong in ra ket qua
# @brief Nhay den function square, lay ket qua va in ra
# @param Ket qua duoc luu o $t0
# @param De in duoc ket so nguyen duong lon thi phai in so nguyen duong khong dau
# @return In ra man hinh ket qua o $a0 neu khong bi tran
# @note Trong he so hoc 32 bit, so nguyen duong lon nhat hien thi duoc la 2^(32)-1
# @note Neu nhap vao so i >= 2^(16) (65536) thi se xay ra hien tuong overflow khi goi ham square(i)
#-------------------------------------------------------------
print_square:
    bgt $a1, 65535, square_over
    blt $a1, -65535, square_over
    jal square_func
    li  $v0, 36
    move $a0, $t0
    syscall
    j   exit_square

square_over:
    la  $a0, msg5
    syscall

exit_square:
    li  $v0, 4
    la  $a0, msg3
    syscall
#-------------------------------------------------------------
# @brief Nhay den chuong trinh chuyen sang he hexa
# @brief In ra "0x"
# @note Vi format in ra so hexa la "0xC" KHONG PHAI la "0x0000000C"
# @note Sau function tim_so_hexa se in truc tiep so hexa su dung mang
#-------------------------------------------------------------
    la  $a0, msg4
    syscall
    jal hex_func
#-------------------------------------------------------------
# @brief Ket thuc chuong trinh
#-------------------------------------------------------------
    li  $v0, 10
    syscall

#-------------------------------------------------------------
# @brief Ham tinh pow(2, i) su dung vong lap va phep dich bit trai de tinh toan
```

```
# @param $t0 luu gia tri i vua la bien dem, vua de lap lai phep dich bit
# @param $t1 vua luu ket qua, vua de dich bit
# @param Su dung vong lap de dich bit, khi $t0 = 0 thi thoat vong lap
# @return Ket qua tra ve o $t1
# @note Chua nghi ra phuong an tim pow(2, i) voi i<0
#-------------------------------------------------------------
pow_func:
    move $t0, $a1
    add  $t1, $0, $0
    addi $t1, $t1, 1
loop:
    beqz $t0, end_pow
    sll  $t1, $t1, 1
    addi $t0, $t0, -1
    bnez $t0, loop
end_pow:
    jr  $ra
#-------------------------------------------------------------
# @brief Ham tinh square su dung phep toan so hoc mulou
# @param Su dung cau lenh mulou de tinh toan so nguyen duong lon khong dau
# @return $t0 chua ket qua can tim
# @note Neu nhap vao so nguyen am thi se duoc chuyen ve so duong de tinh toan
```

```
# @note Chuyen ve so nguyen duong bang cach su dung so bu hai
# @note Ket qua tra ve la so lon nen dung cau lenh mulou
#-------------------------------------------------------------
square_func:
    move $t0, $a1
    bgtz $t0, skip_abs
    nor  $t0, $t0, $0
    addi $t0, $t0, 1
skip_abs:
    mulou $t0, $t0, $t0
end_square:
    jr  $ra
#-------------------------------------------------------------
# @brief Ham chuyen sang he hexa su dung mang luu tru cac ky tu '0'-'9', 'A'-'F'
# @brief Lay i and 0xf de lay noi dung 4-bit cuoi
# @param Khai bao hex la mot mang 8 phan tu de luu cac gia tri ascii cua cac ky tu trong he hexa
# @param $t0 luu gia tri ban dau cua i
# @param $t1 co gia tri la offset cua mang hex, tro den vi tri trong mang hex
# @param $t2 = 0xf
# @param $t5 luu gia tri 4 bit cuoi cua i, cap nhat gia tri trong bang ascii va luu vao mang hex
# @return Mang hex chua cac gia tri ascii sau khi da chuyen sang he hexa
# @return $t1 la gia tri offset cua phan tu cuoi cung co gia tri ascii cua mang hex
```

```
# @note Phan ma gia
# While (i>0) {
#   t5 = i and 0xf
#   if (t5 is digit) t5 += 48
#   else t5+= 55
#   hex[j] = t5
#   i = i >> 4
# }
# @note Vi khi so nhap vao be Vd: 12 thi se phai hien 0xC khong hien 0x0000000C
#-------------------------------------------------------------
hex_func:
    move $t0, $a1
    add  $t1, $0, $0
    addi  $t2, $0, 0xf
outerloop1:
    and  $t5, $t2, $t0
    addi $t5, $t5, 48
    blt  $t5, 58, digit
    addi $t5, $t5, 7
digit:
    sb   $t5, hex($t1)
    addi $t1, $t1, 4
```

```
    srl  $t0, $t0, 4
    beq  $t1, 32, end_hex
    bnez $t0, outerloop1
end_hex:
#-------------------------------------------------------------
# @brief In cac gia tri ascii tu mang hex theo thu tu nguoc lai
# @param $v0 goi service in ky tu trong bang ascii
# @param $t1 tro toi o nho trong mang hex co gia tri va nguoc lai den phan tu dau tien
# @return In ra so hexa
# @note Da in truoc '0x' o phan in truoc do nen chi can in truc tiep so hexa la duoc
#-------------------------------------------------------------
    li   $v0, 11
printloop1:
    lb   $a0, hex($t1)
    syscall
    addi $t1, $t1, -4
    bgez $t1, printloop1
    jr   $ra
```