

Mini Projects

Lớp: 139361 – Học phần: Thực hành Kiến trúc máy tính

Họ và tên: Nguyễn Cao Kỳ

MSSV: 20215072

Họ và tên: Đoàn Văn Linh

MSSV: 20210531

Project 10

```
.data
Message: .asciiz "Nhap so nguyen:"
Title: .asciiz "i power(2,i) square(i) Hexadecimal(i)\n"
space: .asciiz " "
digit: .asciiz "0123456789ABCDEF"
hex_result: .asciiz "0x"

.text
main:
    li $v0, 51
    la $a0, Message
    syscall
    add $s0, $zero, $a0
    nop
    jal convert_powerOfInput
    add $s1, $zero, $v0
    nop
    jal convert_squared
    add $s2, $zero, $v0
    nop
    jal convert_hex
    add $s3, $zero, $v0
    nop

    la $a0, Title
    li $v0, 4
    syscall
    add $a0, $zero, $s0
    add $a1, $zero, $s1
    add $a2, $zero, $s2
    add $a3, $zero, $s3
    jal print_table
    nop
    li $v0, 10
    syscall
endmain:

#-----
# function convert_powerOfInput
# param[in] $a0 the exponent
# return $v0 result of 2 raised to the power of input
#-----
convert_powerOfInput:
    beq $a0, $zero, case_zero
```

```

    addi    $v0, $zero, 2
    addi    $t0, $zero, 1

loop:
    beq $a0, $t0, done_powerOfInput
    sll $v0, $v0, 1
    addi    $t0, $t0, 1
    j      loop
case_zero:
    addi    $v0, $zero, 1
done_powerOfInput:
    jr $ra

#-----
# function convert_squared
# param[in] $a0 the base
# return $v0 result of squared base
#-----
convert_squared:
    mul $v0, $a0, $a0
done_squared:
    jr $ra

#-----
# function convert_hexa
# param[in] $a0 input integer
# return $v0 the address of the array storing hexa value in string type
#-----
convert_hex:
    la $t0, hex_result
    addi $t0, $t0, 2
    la $t1, digit
    li $t2, 8          # total loops

loop_hex:
    andi $t3, $a0, 0xf          # retrieve LSB
    add $t4, $t0, $t2          # byte-saving address

    add $t5, $t1, $t3          # digit address
    lb $t3, 0($t5)             # retrieve digit element
    sb $t3, 0($t4)
    addi $t2, $t2, -1
    srl $a0, $a0, 4
    beq $a0, $zero, get_hex
    j loop_hex
get_hex:
    add $v0, $zero, $t4
    addi $v0, $v0, -1

    addi $t0, $zero, 120
    sb $t0, 0($v0)

    addi $v0, $v0, -1
    addi $t0, $zero, 48
    sb $t0, 0($v0)

end_loop_hex:

```

```
done_hex:
    jr $ra
```

```
#-----
# function print_table
# param[in] $a0 i
# param[in] $a1 2^i
# param[in] $a2 i squared
# param[in] $a3 hexa value of i
# return table
#-----
```

```
print_table:
```

```
    li $v0, 1
    syscall
```

```
    nop
```

```
    la $a0, space
    li $v0, 4
    syscall
```

```
    nop
```

```
    add $a0, $zero, $s1
    li $v0, 1
    syscall
```

```
    nop
```

```
    la $a0, space
    li $v0, 4
    syscall
```

```
    nop
```

```
    la $a0, space
    li $v0, 4
    syscall
```

```
    nop
```

```
    add $a0, $zero, $s2
    li $v0, 1
    syscall
```

```
    nop
```

```
    la $a0, space
    li $v0, 4
    syscall
```

```
    nop
```

```
    la $a0, space
    li $v0, 4
    syscall
```

```
    nop
```

```
    add $a0, $zero, $s3
    li $v0, 4
    syscall
```

```
done_print_table:
```

```
jr $ra
```

Project 10 yêu cầu nhập một số nguyên i và in ra các giá trị 2^i , i^2 và dạng hexadecimal của i . Các kết quả trên được tính thông qua các procedure `convert_powerOfInput`, `convert_squared` và `convert_hex`, trong khi bảng sẽ được in ra bằng procedure `print_table`.

Để bắt đầu, nhập i để tiến hành tính toán. Gán $\$v0 = 51$ và nhập vào số nguyên. Lưu số nguyên vào $\$s0$.

The screenshot shows the MARS 4.5 IDE interface. The main window displays assembly code for a program that calculates powers and squares of an input value. The code includes labels for `main`, `endmain`, `convert_powerOfInput`, `case_zero`, `done_powerOfInput`, `convert_squared`, `done_squared`, `convert_hex`, and `loop_hex`. The registers window on the right shows the state of various registers, with `$a0` highlighted. An input dialog box is open, prompting the user to enter a value for `Nhap so nguyen:`. The bottom panel shows the output of the program, displaying the results of the calculations for `power(2,i)`, `square(i)`, and `Hexadecimal(i)`.

Register	Value
\$zero	0x00000000
\$at	0x10010000
\$v0	0x00000033
\$v1	0x00000000
\$a0	0x10010000
\$a1	0x00000000
\$a2	0x00000000
\$a3	0x00000000
\$t0	0x00000000
\$t1	0x00000000
\$t2	0x00000000
\$t3	0x00000000
\$t4	0x00000000
\$t5	0x00000000
\$t6	0x00000000
\$t7	0x00000000
\$s0	0x00000000
\$s1	0x00000000
\$s2	0x00000000
\$s3	0x00000000
\$s4	0x00000000
\$s5	0x00000000
\$s6	0x00000000
\$s7	0x00000000
\$s8	0x00000000
\$s9	0x00000000
\$k0	0x00000000
\$k1	0x00000000
\$gp	0x10008000
\$sp	0x7ffffeff
\$fp	0x00000000
\$ra	0x00000000
pc	0x0040000e
hi	0x00000000
lo	0x00000000

Tiến hành tính 2^i .

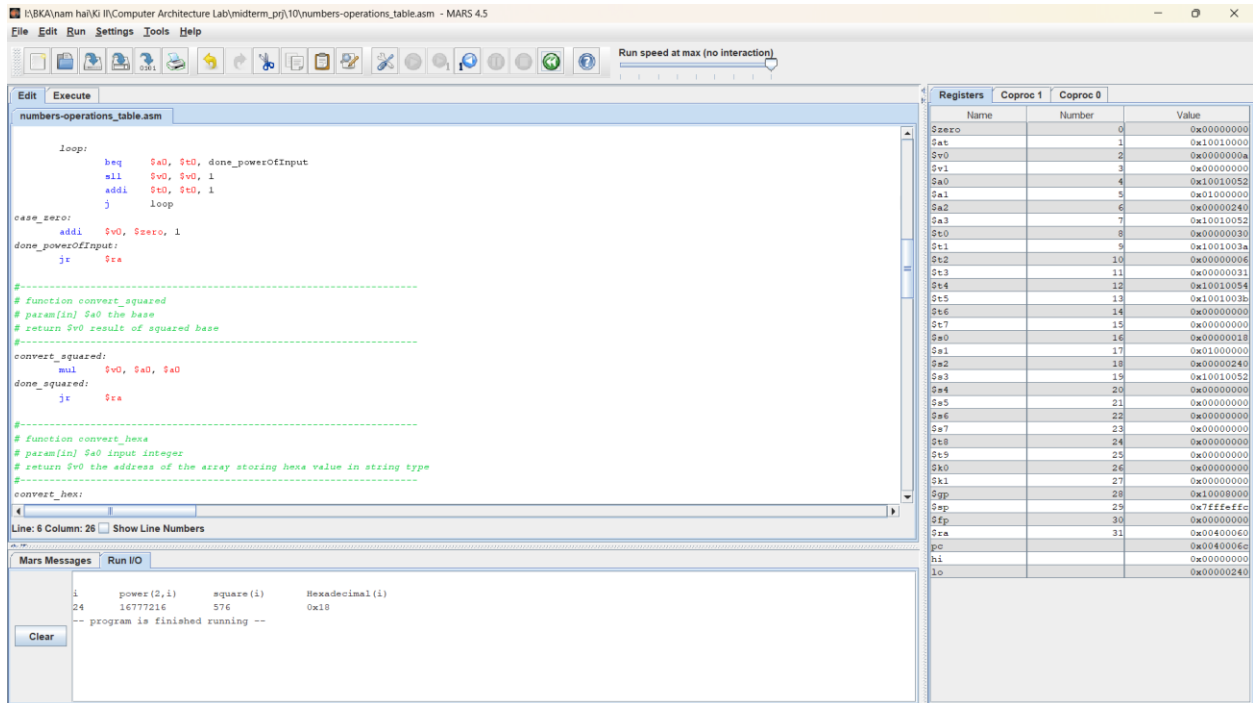
Hàm có đầu vào \$a0 là số mũ của 2^i , hay chính là số nguyên vừa nhập.

Đầu tiên, kiểm tra nếu \$a0 = 0, nhảy đến case_zero, gán \$v0 = 1 và nhảy đến done_powerOfInput, kết thúc procedure.

Trong trường hợp này, vì \$a0 khác 0, tiếp theo thực hiện gán giá trị ban đầu \$v0 = 2 và số lần dịch bit ban đầu \$t0 là 1. Nhảy đến vòng lặp. Kiểm tra nếu số lần dịch bit hiện tại bằng \$a0 thì kết thúc vòng lặp. Ngược lại, thực hiện dịch \$v0 1 bit, tăng \$t0 lên 1 đơn vị và lặp lại vòng lặp cho đến khi thỏa mãn yêu cầu.

Sau đó, lưu kết quả vào \$s1.

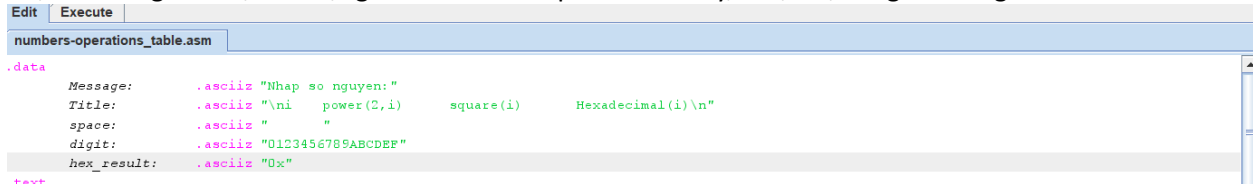
Tiếp theo, thực hiện tính i^2 thông qua procedure `convert_squared`. Tham số đầu vào \$a0 là số nguyên đã nhập.

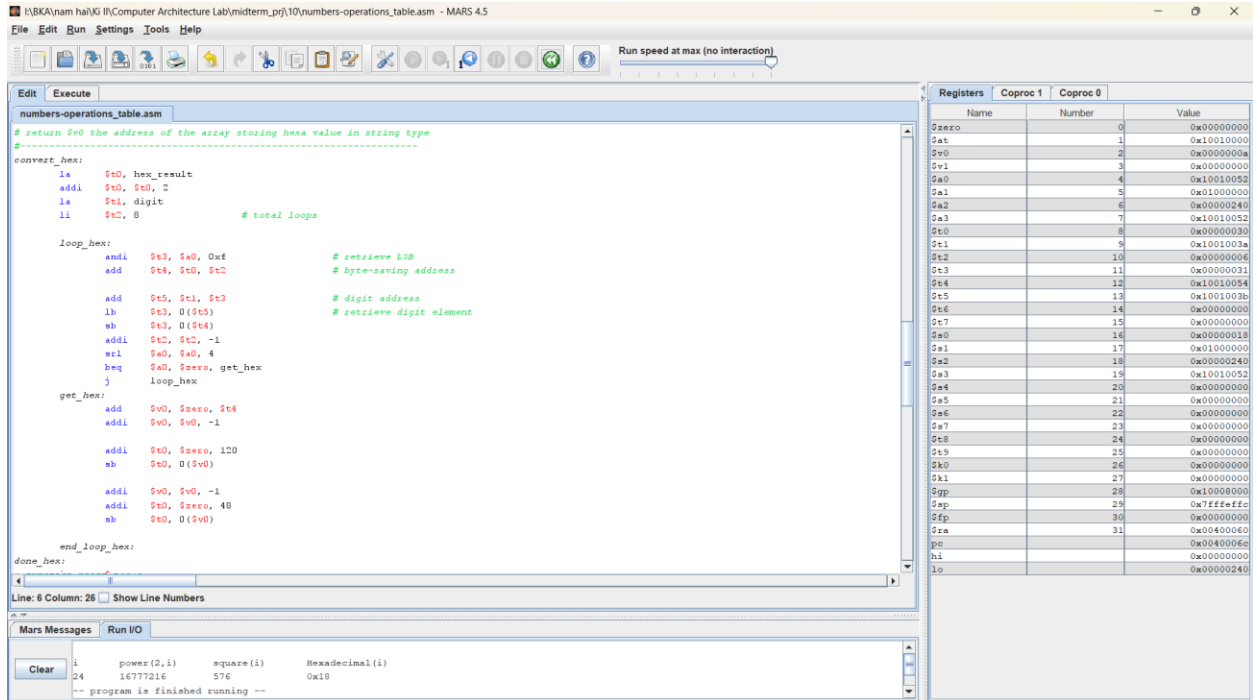


Bắt đầu, thực hiện phép nhân \$a0 với \$a0 và lưu vào \$v0. Kết thúc procedure. Lưu kết quả vào \$s1.

Cuối cùng, thực hiện chuyển đổi số nguyên đã nhập thành dạng hexadecimal thông qua procedure `convert_hex`. Tham số đầu vào vẫn là số nguyên đã nhập và sẽ trả về chuỗi ghi giá trị của số nguyên ở hệ 16.

Một số mảng sẽ được sử dụng để hoàn thành procedure này, được định nghĩa trong `.data`.





Đầu tiên, lưu địa chỉ của mảng `hex_result` vào `$t0`. Vì `hex_result` bắt đầu bằng chuỗi “0x”, thêm 2 đơn vị vào `$t0` để tiện cho việc tính toán. Lưu địa chỉ mảng tham chiếu `digit` vào `$t1` và tổng số lần lấy bit nhỏ nhất `$t2` là 8.

Vòng lặp có 2 giai đoạn: lấy bit nhỏ nhất và lưu bit đó vào mảng.

Giai đoạn một bắt đầu bằng việc lấy bit nhỏ nhất, sử dụng lệnh

```
andi    $t3, $a0, 0xf,
```

lưu vào `$t3`. Sau đó tính địa chỉ cần lưu bit nhỏ nhất chính là `$t0 + $t2` lưu vào `$t4` và lấy ra kí tự tương đương với bit nhỏ nhất đó dựa vào mảng tham chiếu.

Mảng tham chiếu `digit` có dạng “0123456789ABCDEF”, do đó, `$t5 = $t1 + $t3` sẽ luôn lấy ra được địa chỉ của kí tự tương đương.

Giai đoạn tiếp theo, sau khi đã lấy ra được địa chỉ kí tự tương đương và lưu vào `$t5`, thực hiện load byte giá trị địa chỉ đó vào `$t3`, sau đó store byte giá trị đó vào `$t4`. Cuối cùng, thực hiện trừ `$t2` đi 1 đơn vị và dịch `$a0` đi 4 bit để loại đi bit đã lấy ra. Kiểm tra nếu `$a0 = 0`, nhảy đến `get_hex`. Ngược lại, tiếp tục vòng lặp.

Tại `get_hex`, ta đã có tất cả các bit được lấy ra. Khi đó, địa chỉ phần tử cuối cùng được lưu vào nằm ở `$t4`. Do đó lần lượt trừ `$t4` đi 1 đơn vị và store byte hai ký tự “x” và “0”. Cuối cùng đã có chuỗi hexadecimal hoàn chỉnh. Lưu địa chỉ `$t4` vào `$v0`, kết thúc procedure.

Cuối cùng, sử dụng procedure `print_table` để in kết quả, sử dụng `syscall` để in chuỗi và số nguyên vào I/O.

Kết quả:

Mars Messages

Run I/O

```

i      power(2,i)      square(i)      Hexadecimal(i)
24    16777216        576            0x18
-- program is finished running --

```

Clear

```

i      power(2,i)      square(i)      Hexadecimal(i)
24    16777216        576            0x18
-- program is finished running --

```

Data segment:

Data Segment									
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)	
0x10010000	p a h N	o s	y u g n	\0 : n e	p \t i \n	r e w o	i , 2 (q s \t)	▲
0x10010020	e r a u	\t) i (a x e H	i c e d	(l a m	\0 \n) i	1 0 \0 \t	5 4 3 2	
0x10010040	9 8 7 6	D C B A	0 \0 F E	\0 \0 \0 x	x 0 \0 \0	\0 \0 8 1	\0 \0 \0 \0	\0 \0 \0 \0	
0x10010060	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	
0x10010080	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	
0x100100a0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	
0x100100c0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	
0x100100e0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	
0x10010100	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	
0x10010120	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	\0 \0 \0 \0	▼

Registers:

Registers	Coproc 1	Coproc 0
Name	Number	Value
\$zero	0	0x00000000
\$at	1	0x10010000
\$v0	2	0x0000000a
\$v1	3	0x00000000
\$a0	4	0x10010052
\$a1	5	0x01000000
\$a2	6	0x00000240
\$a3	7	0x10010052
\$t0	8	0x00000030
\$t1	9	0x1001003a
\$t2	10	0x00000006
\$t3	11	0x00000031
\$t4	12	0x10010054
\$t5	13	0x1001003b
\$t6	14	0x00000000
\$t7	15	0x00000000
\$s0	16	0x00000018
\$s1	17	0x01000000
\$s2	18	0x00000240
\$s3	19	0x10010052
\$s4	20	0x00000000
\$s5	21	0x00000000
\$s6	22	0x00000000
\$s7	23	0x00000000
\$t8	24	0x00000000
\$t9	25	0x00000000
\$k0	26	0x00000000
\$k1	27	0x00000000
\$gp	28	0x10008000
\$sp	29	0x7fffeffc
\$fp	30	0x00000000
\$ra	31	0x00400060
pc		0x0040006c
hi		0x00000000
lo		0x00000240

Project 21

```

.data
msgInput:  .ascii "Enter a number"
msgResult: .ascii "Digit degree is "
.text
main:
    la $a0, msgInput
    li $v0, 51
    syscall
    add $s0, $zero, $a0

    add $a0, $zero, $s0

```

```

        jal calculate_digit_degree
        add $s1, $zero, $v0
        nop

        la $a0, msgResult
        add $a1, $zero, $v0
        li $v0, 56
        syscall
        nop

        li $v0, 10
        syscall
    endmain:
#-----
# function calculate_digit_degree
# param[in] $a0 input number
# return $v0 digit degree
#-----
calculate_digit_degree:
    xor $t0, $zero, $zero    # Digit degree = 0
    addi $t1, $zero, 10
    blt $a0, $t1, end_digit_degree

    set_sum:
        xor $t2, $zero, $zero    # sum = 0
    sumOfDigits_loop:
        div $a0, $t1
        mfhi $t3
        add $t2, $t2, $t3
        mflo $a0
        beq $a0, $zero, check_end
        j sumOfDigits_loop
    check_end:
        addi $t0, $t0, 1
        blt $t2, $t1, end_digit_degree
        add $a0, $zero, $t2
        j set_sum

    end_digit_degree:
        add $v0, $zero, $t0
        jr $ra

```

Project 21 yêu cầu nhập vào 1 số nguyên, và tính digit degree của số nguyên đó. Ở đây, việc đó sẽ được tính toán thông qua procedure `calculate_digit_degree`.

Bước đầu tiên, tương tự như trên, gán `$v0 = 51` và nhập vào số nguyên. Lưu số nguyên vào `$s0`.

h\BKA\nam ha\KG I\Computer Architecture Lab\midterm_prj\21\digit_degree.asm - MARS 4.5

File Edit Run Settings Tools Help

Run speed at max (no interaction)

Edit Execute

Text Segment

Bkpt	Address	Code	Basic	Source
0x00400000	0x3c011001	lui	\$1,0x00001001	6: la \$a0, msgInput
0x00400004	0x34240000	ori	\$4,\$1,0x00000000	
0x00400008	0x24020033	addiu	\$2,\$0,0x0000...	7: li \$r0, \$1
0x0040000c	0x0000000c	syscall		8: syscall
0x00400010	0x00480200	add	\$16,\$0,\$4	9: add \$a0, \$zero, \$a0
0x00400014	0x00102020	add	\$4,\$0,\$16	11: add \$a0, \$zero, \$a0
0x00400018	0x00100011	jal	0x00400044	12: jal calculate_digit_degree
0x0040001c	0x00280200	add	\$17,\$0,\$2	13: add \$a1, \$zero, \$v0
0x00400020	0x00000000	nop		14: nop
0x00400024	0x3c011001	lui	\$1,0x00001001	16: la \$a0, msgResult
0x00400028	0x34240000	ori	\$4,\$1,0x00000000	
0x0040002c	0x00022820	add	\$5,\$0,\$2	17: add \$a1, \$zero, \$v0

Labels

Label	Address
digit_degree.asm	
main	0x00400000
endmain	0x00400044
calculate_digit_degree	0x00400044
set_sum	0x00400054
sumOfDigits_loop	0x00400058
check_end	0x00400070
end_digit_degree	0x00400084
msgInput	0x10010000
msgResult	0x1001000F

Registers

Name	Number	Value
\$zero	0	0x00000000
\$at	1	0x10010000
\$v0	2	0x00000033
\$v1	3	0x00000000
\$a0	4	0x10010000
\$a1	5	0x00000000
\$a2	6	0x00000000
\$a3	7	0x00000000
\$t0	8	0x00000000
\$t1	9	0x00000000
\$t2	10	0x00000000
\$t3	11	0x00000000
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$s0	16	0x00000000
\$s1	17	0x00000000
\$s2	18	0x00000000
\$s3	19	0x00000000
\$s4	20	0x00000000
\$s5	21	0x00000000
\$s6	22	0x00000000
\$s7	23	0x00000000
\$t8	24	0x00000000
\$t9	25	0x00000000
\$k0	26	0x00000000
\$k1	27	0x00000000
\$gp	28	0x10008000
\$fp	29	0x7ffffeffc
\$ra	30	0x00000000
\$pc	31	0x00400000
hi		0x00000000
lo		0x00000000

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)
0x10010000	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
0x10010020	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
0x10010040	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
0x10010060	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
0x10010080	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
0x100100A0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
0x100100C0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
0x100100E0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
0x10010100	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
0x10010120	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0

Mars Messages

Run I/O

Clear

i power(2,i) square(i) Hexadecimal(i)
24 16777216 576 0x18
-- program is finished running --

h\BKA\nam ha\KG I\Computer Architecture Lab\midterm_prj\21\digit_degree.asm - MARS 4.5

File Edit Run Settings Tools Help

Run speed at max (no interaction)

Edit Execute

Text Segment

Bkpt	Address	Code	Basic	Source
0x00400000	0x3c011001	lui	\$1,0x00001001	6: la \$a0, msgInput
0x00400004	0x34240000	ori	\$4,\$1,0x00000000	
0x00400008	0x24020033	addiu	\$2,\$0,0x0000...	7: li \$r0, \$1
0x0040000c	0x0000000c	syscall		8: syscall
0x00400010	0x00480200	add	\$16,\$0,\$4	9: add \$a0, \$zero, \$a0
0x00400014	0x00102020	add	\$4,\$0,\$16	11: add \$a0, \$zero, \$a0
0x00400018	0x00100011	jal	0x00400044	12: jal calculate_digit_degree
0x0040001c	0x00280200	add	\$17,\$0,\$2	13: add \$a1, \$zero, \$v0
0x00400020	0x00000000	nop		14: nop
0x00400024	0x3c011001	lui	\$1,0x00001001	16: la \$a0, msgResult
0x00400028	0x34240000	ori	\$4,\$1,0x00000000	
0x0040002c	0x00022820	add	\$5,\$0,\$2	17: add \$a1, \$zero, \$v0

Labels

Label	Address
digit_degree.asm	
main	0x00400000
endmain	0x00400044
calculate_digit_degree	0x00400044
set_sum	0x00400054
sumOfDigits_loop	0x00400058
check_end	0x00400070
end_digit_degree	0x00400084
msgInput	0x10010000
msgResult	0x1001000F

Registers

Name	Number	Value
\$zero	0	0x00000000
\$at	1	0x10010000
\$v0	2	0x00000033
\$v1	3	0x00000000
\$a0	4	0x00005eff4
\$a1	5	0x00000000
\$a2	6	0x00000000
\$a3	7	0x00000000
\$t0	8	0x00000000
\$t1	9	0x00000000
\$t2	10	0x00000000
\$t3	11	0x00000000
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$s0	16	0x00005eff4
\$s1	17	0x00000000
\$s2	18	0x00000000
\$s3	19	0x00000000
\$s4	20	0x00000000
\$s5	21	0x00000000
\$s6	22	0x00000000
\$s7	23	0x00000000
\$t8	24	0x00000000
\$t9	25	0x00000000
\$k0	26	0x00000000
\$k1	27	0x00000000
\$gp	28	0x10008000
\$fp	29	0x7ffffeffc
\$ra	30	0x00000000
\$pc	31	0x00400000
hi		0x00000000
lo		0x00000000

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
0x10010020	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
0x10010040	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
0x10010060	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
0x10010080	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
0x100100A0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
0x100100C0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
0x100100E0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
0x10010100	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
0x10010120	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0

Mars Messages

Run I/O

Clear

i power(2,i) square(i) Hexadecimal(i)
24 16777216 576 0x18
-- program is finished running --

Tiếp theo, gán \$a0 = \$s0 để tạo tham số đầu vào cho procedure sẽ được sử dụng.

File Edit Run Settings Tools Help

Run speed at max (no interaction)

Text Segment

Bkpt	Address	Code	Basic	Source
0x00400000	0x3e011001	lui	\$1,0x00001001	6: la \$a0, msgInput
0x00400004	0x34240000	ori	\$4,\$1,0x00000000	
0x00400008	0x24020033	addi	\$2,\$2,\$2,0x00000000	7: li \$r0, \$1
0x0040000c	0x00000000	syscall		8: syscall
0x00400010	0x00480200	add	\$16,\$0,\$4	9: add \$a0, \$zero, \$a0
0x00400014	0x00102020	add	\$4,\$0,\$16	11: add \$a0, \$zero, \$a0
0x00400018	0x00100011	jal	0x00400044	12: jal calculate_digit_degree
0x0040001c	0x00208020	add	\$17,\$0,\$2	13: add \$a1, \$zero, \$v0
0x00400020	0x00000000	nop		14: nop
0x00400024	0x3e011001	lui	\$1,0x00001001	16: la \$a0, msgResult
0x00400028	0x34240000	ori	\$4,\$1,0x00000000	
0x0040002c	0x00022820	add	\$5,\$0,\$2	17: add \$a1, \$zero, \$v0

Labels

Label	Address
digit_degree.asm	
main	0x00400000
endmain	0x00400044
calculate_digit_degree	0x00400044
set_sum	0x00400054
sumOfDigits_loop	0x00400058
check_end	0x00400070
end_digit_degree	0x00400084
msgInput	0x10010000
msgResult	0x10010005

Registers

Name	Number	Value
\$zero	0	0x00000000
\$at	1	0x10010000
\$v0	2	0x00000033
\$v1	3	0x00000000
\$a0	4	0x00000000
\$a1	5	0x00000000
\$a2	6	0x00000000
\$a3	7	0x00000000
\$t0	8	0x00000000
\$t1	9	0x00000000
\$t2	10	0x00000000
\$t3	11	0x00000000
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$a0	16	0x00000000
\$a1	17	0x00000000
\$a2	18	0x00000000
\$a3	19	0x00000000
\$t0	20	0x00000000
\$t1	21	0x00000000
\$t2	22	0x00000000
\$t3	23	0x00000000
\$t4	24	0x00000000
\$t5	25	0x00000000
\$t6	26	0x00000000
\$t7	27	0x00000000
\$gp	28	0x10008000
\$sp	29	0x7ffffeffc
\$fp	30	0x00000000
\$ra	31	0x00000000
\$pc		0x00400018
\$hi		0x00000000
\$lo		0x00000000

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	a	t	n	e				
0x10010020	\0	\0	\0	\0	\0	\0	\0	\0
0x10010040	\0	\0	\0	\0	\0	\0	\0	\0
0x10010060	\0	\0	\0	\0	\0	\0	\0	\0
0x10010080	\0	\0	\0	\0	\0	\0	\0	\0
0x100100a0	\0	\0	\0	\0	\0	\0	\0	\0
0x100100c0	\0	\0	\0	\0	\0	\0	\0	\0
0x100100e0	\0	\0	\0	\0	\0	\0	\0	\0
0x10010100	\0	\0	\0	\0	\0	\0	\0	\0
0x10010120	\0	\0	\0	\0	\0	\0	\0	\0

Mars Messages

```

i      power(2,i)      square(i)      Hexadecimal(i)
24    16777216        576            0x10
-- program is finished running --

```

File Edit Run Settings Tools Help

Run speed at max (no interaction)

numbers-operations_table.asm mips2.asm digit_degree.asm

```

25 #-----
26 # function calculate_digit_degree
27 # param[in] $a0 input number
28 # return $v0 digit degree
29 #-----
30 calculate_digit_degree:
31     xor $t0, $zero, $zero      # Digit degree = 0
32     addi $t1, $zero, 10
33     bgt $a0, $t1, end_digit_degree
34
35     set_sum:
36     xor $t2, $zero, $zero      # sum = 0
37     sumOfDigits_loop:
38     div $a0, $t1
39     mthi $t3
40     add $t2, $t2, $t3
41     mflo $a0
42     beq $a0, $zero, check_end
43     j sumOfDigits_loop
44     check_end:
45     addi $t0, $t0, 1
46     bgt $t2, $t1, end_digit_degree
47     add $a0, $zero, $t2
48     j set_sum
49
50 end_digit_degree:
51     add $v0, $zero, $t0
52     jr $ra
53
54
55
56

```

Registers

Name	Number	Value
\$zero	0	0x00000000
\$at	1	0x00000000
\$v0	2	0x00000000
\$v1	3	0x00000000
\$a0	4	0x00000000
\$a1	5	0x00000000
\$a2	6	0x00000000
\$a3	7	0x00000000
\$t0	8	0x00000000
\$t1	9	0x00000000
\$t2	10	0x00000000
\$t3	11	0x00000000
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$a0	16	0x00000000
\$a1	17	0x00000000
\$a2	18	0x00000000
\$a3	19	0x00000000
\$t0	20	0x00000000
\$t1	21	0x00000000
\$t2	22	0x00000000
\$t3	23	0x00000000
\$t4	24	0x00000000
\$t5	25	0x00000000
\$t6	26	0x00000000
\$t7	27	0x00000000
\$gp	28	0x10008000
\$sp	29	0x7ffffeffc
\$fp	30	0x00000000
\$ra	31	0x00000000
\$pc		0x00400018
\$hi		0x00000000
\$lo		0x00000000

Mars Messages

```

i      power(2,i)      square(i)      Hexadecimal(i)
24    16777216        576            0x10
-- program is finished running --

```

Procedure có tham số đầu vào là \$a0 chính là số nguyên vừa nhập.

Đầu tiên, khởi tạo \$t0 là digit degree ban đầu. Kiểm tra nếu \$a0 < 10, nhảy đến end_digit_degree.

Lưu giá trị \$t0 vào \$v0 và kết thúc procedure.

Nếu \$a0 >= 10, bắt đầu set_sum. Ở đây, gán \$t2 = 0 chính là tổng hiện tại của tất cả chữ số của \$a0. Bắt đầu sumOfDigits_loop. Vòng lặp thực hiện lấy ra chữ số cuối cùng của \$a0 bằng cách thực hiện phép chia cho 10 và lấy phần dư. Thực hiện phép chia \$a0 và \$t1. Phần dư được lưu vào thanh ghi \$hi và phần nguyên lưu vào thanh ghi \$lo. Ta lấy ra phần dư, cộng vào \$t2 và gán \$a0 bằng phần nguyên. Lúc này, nếu \$a0 = 0, nhảy đến check_end. Ngược lại, quay lại vòng lặp.

Tại check_end, đầu tiên tăng \$t0 lên 1 đơn vị, biểu thị đã kết thúc 1 lần tính tổng. Kiểm tra nếu tổng \$t2 lúc này nhỏ hơn 10, nhảy đến end_digit_degree để kết thúc procedure. Ngược lại, gán \$a0 = \$t2 và quay lại vòng lặp. Vòng lặp được lặp lại cho đến khi thỏa mãn yêu cầu.

Sau khi kết thúc procedure, kết quả sẽ được in ra thông qua một dialog.

Kết quả:

The screenshot shows the MARS 4.5 IDE interface. The main window displays the assembly code for 'digit_degree.asm'. A dialog box in the center shows 'Digit degree is 2'. The right panel shows the register values, with \$a0 highlighted at 4. The bottom panel shows the Mars Messages window with the output of the program.

Register	Value
\$zero	0x00000000
\$at	0x10010000
\$v0	0x00000000
\$v1	0x00000000
\$a0	0x00000004
\$a1	0x00000000
\$a2	0x00000000
\$a3	0x00000000
\$t0	0x00000000
\$t1	0x00000000
\$t2	0x00000000
\$t3	0x00000000
\$t4	0x00000000
\$t5	0x00000000
\$t6	0x00000000
\$t7	0x00000000
\$s0	0x00000000
\$s1	0x00000000
\$s2	0x00000000
\$s3	0x00000000
\$s4	0x00000000
\$s5	0x00000000
\$s6	0x00000000
\$s7	0x00000000
\$s8	0x00000000
\$s9	0x00000000
\$k0	0x00000000
\$k1	0x00000000
\$gp	0x10010000
\$fp	0x7ffff000
\$fr	0x00000000
\$ra	0x00000000
\$pc	0x00400018
\$hi	0x00000000
\$lo	0x00000000

Mars Messages:

```

i      power(2,i)      square(i)      Hexadecimal(i)
24    16777216         576            0x10
-- program is finished running --

```

Registers:

Registers	Coproc 1	Coproc 0	
Name	Number	Value	
\$zero	0	0x00000000	
\$at	1	0x10010000	
\$v0	2	0x0000000a	
\$v1	3	0x00000000	
\$a0	4	0x1001000f	
\$a1	5	0x00000002	
\$a2	6	0x00000000	
\$a3	7	0x00000000	
\$t0	8	0x00000002	
\$t1	9	0x0000000a	
\$t2	10	0x00000003	
\$t3	11	0x00000002	
\$t4	12	0x00000000	
\$t5	13	0x00000000	
\$t6	14	0x00000000	
\$t7	15	0x00000000	
\$s0	16	0x00005ff4	
\$s1	17	0x00000002	
\$s2	18	0x00000000	
\$s3	19	0x00000000	
\$s4	20	0x00000000	
\$s5	21	0x00000000	
\$s6	22	0x00000000	
\$s7	23	0x00000000	
\$t8	24	0x00000000	
\$t9	25	0x00000000	
\$k0	26	0x00000000	
\$k1	27	0x00000000	
\$gp	28	0x10008000	
\$sp	29	0x7fffeffc	
\$fp	30	0x00000000	
\$ra	31	0x0040001c	
pc		0x00400044	
hi		0x00000002	
lo		0x00000000	