LẬP TRÌNH ASSEMBLY

Đỗ Thanh Nghị
dtnghi@cit.ctu.edu.vn

12-2019

Giới thiệu lập trình assembly

- Hợp ngữ (assembly language) và kiến trúc máy tính
- CISC (complex instruction set computer): x86
- RISC (reduced instruction set computer): ARM, MIPS
- Lập trình cho kiến trúc x86
- Microsoft Assembler (MASM)
- Borland Turbo Assembler (TASM)
- The GNU assembler (GAS)
- The Netwide Assembler (**NASM**)

Chương trình assembly

- Chương trình có 3 phần: data, bss, text

- data: khai báo dữ liệu khởi tạo, hằng

section .data

- bss: khai báo biến

section .bss

- text: bắt đầu chương trình, khai báo biến toàn cục _start để kernel biết đầu chương trình

section .text

hello.asm


```
section .text
   global _start      ;must be declared for linker (ld)

_start:               ;tells linker entry point
   mov  edx,len       ;message length
   mov  ecx,msg       ;message to write
   mov  ebx,1         ;file descriptor (stdout)
   mov  eax,4         ;system call number (sys_write)
   int  0x80          ;call kernel

   mov  eax,1         ;system call number (sys_exit)
   int  0x80          ;call kernel

section .data
msg db 'Hello, world!', 0xa  ;string to be printed
len equ $ - msg      ;length of the string
```

Hợp dịch, liên kết

- Hợp dịch tập tin .asm => .o

nasm -f elf32 hello.asm (-f elf64)

Nếu không có lỗi => hello.o

- Liên kết

ld -m elf_i386 -s -o hello hello.o (-m elf_x86_64)

- Thực thi

./hello

Hello, world!

Hợp dịch, liên kết

- Lập trình online

https://www.tutorialspoint.com/compile_assembly_online.php

Execute | Share | main.asm | STDIN

```
section .text
    global _start       ;must be declared for using
        gcc
_start:                 ;tell linker entry
    point
    mov edx, len    ;message length
    mov ecx, msg    ;message to write
    mov ebx, 1      ;file descriptor (stdout)
    mov eax, 4      ;system call number (sys_write)
    int 0x80        ;call kernel
    mov eax, 1      ;system call number (sys_exit)
    int 0x80        ;call kernel

section .data

msg db  'Hello, world!',0xa ;our dear string
len equ $ - msg         ;length of our dear string
```

Result

```
$nasm -f elf *.asm; ld -m elf_i386 -s -o demo *.o
$demo
Hello, world!
```

Bộ nhớ

high memory

stack
.
.
.
heap
BSS – uninitialized data
data
text (code)
reserved

low memory

Các phân đoạn

- Có thể thay thế section bằng các segment
- data segment: section .data và section .bss
- code segment: section .text
- stack: chứa dữ liệu truyền cho hàm, thủ tục

Lập trình assembly


```
segment .text        ;code segment
   global_start      ;must be declared for linker

_start:              ;tell linker entry point
   mov edx,len       ;message length
   mov ecx,msg       ;message to write
   mov ebx,1         ;file descriptor (stdout)
   mov eax,4         ;system call number (sys_write)
   int 0x80          ;call kernel

   mov eax,1         ;system call number (sys_exit)
   int 0x80          ;call kernel

segment .data        ;data segment
msg     db 'Hello, world!',0xa    ;our dear string
len     equ     $ - msg           ;length of our dear string
```

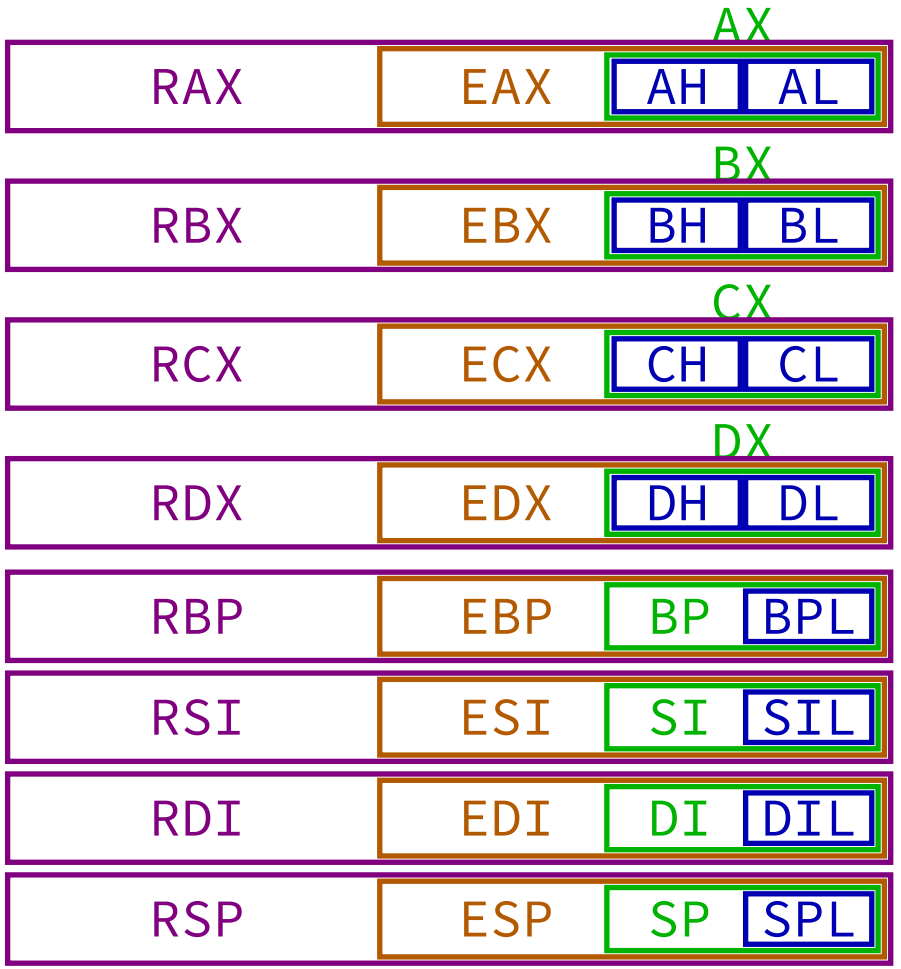
Các thanh ghi

- Thanh ghi tổng quát
  - Dữ liệu - 16 bits: AX, BX, CX, DX; 32 bits: EAX, EBX, ECX, EDX; 64 bits: RAX, RBX, RCX, RDX

  16 bits => 32 bits (**E**xtended) => 64 bits (**R**egister)
  - Con trỏ - lệnh: IP, EIP, RIP; ngăn xếp: SP, ESP, RSP; nền: BP, EBP, RBP
  - Chỉ mục - nguồn: SI, ESI, RSI; đích: DI, EDI, RDI
- Thanh ghi điều khiển: Overflow Flag (OF), Zero Flag (ZF), Sign Flag (SF), Carry Flag (CF), Parity Flag (PF), Direction Flag (DF), Interrupt Flag (IF), Trap Flag (TF),
- Thanh ghi đoạn: Code Segment (CS), Data segment (DS), Stack Segment (SS)

Các thanh ghi

- 16 bits: AX, BX, CX, DX, BP, SI, DI, SP
- 32 bits: EAX, EBX, ECX, EDX, EBP, ESI, EDI, ESP
- 64 bits: RAX, RBX, RCX, RDX, RBP, RSI, RDI, RSP



Các thanh ghi


```
section .text
   global _start     ;must be declared for linker (gcc)

_start:              ;tell linker entry point
   mov  edx,len   ;message length
   mov  ecx,msg   ;message to write
   mov  ebx,1     ;file descriptor (stdout)
   mov  eax,4     ;system call number (sys_write)
   int  0x80      ;call kernel

   mov  edx,9     ;message length
   mov  ecx,s2    ;message to write
   mov  ebx,1     ;file descriptor (stdout)
   mov  eax,4     ;system call number (sys_write)
   int  0x80      ;call kernel

   mov  eax,1     ;system call number (sys_exit)
   int  0x80      ;call kernel

section .data
msg db 'Displaying 9 stars',0xa ;a message
len equ $ - msg  ;length of message
s2 times 9 db '*'
```

Lời gọi hệ thống Linux

- Thực hiện lời gọi hệ thống
  - Đặt số hiệu lời gọi vào EAX
  - Tham số cho lời gọi trong EBX, ECX, etc
  - Gọi ngắt 80h
  - Kết quả trả về trong EAX

%eax	Name	%ebx	%ecx	%edx	%esx	%edi
1	sys_exit	int	-	-	-	-
2	sys_fork	struct pt_regs	-	-	-	-
3	sys_read	unsigned int	char *	size_t	-	-
4	sys_write	unsigned int	const char *	size_t	-	-
5	sys_open	const char *	int	int	-	-
6	sys_close	unsigned int	-	-	-	-

Lời gọi hệ thống Linux: ví dụ

Thoát khỏi hệ thống

```
mov     eax,1               ; system call number (sys_exit)
int     0x80                ; call kernel
```

Ghi chuỗi msg (dài 4) ra màn hình

```
mov     edx,4               ; message length
mov     ecx,msg             ; message to write
mov     ebx,1               ; file descriptor (stdout)
mov     eax,4               ; system call number (sys_write)
int     0x80                ; call kernel
```

Lời gọi hệ thống Linux: ví dụ


```
section .data                              ;Data segment
   userMsg db 'Please enter a number: ' ;Ask the user to enter a number
   lenUserMsg equ $-userMsg                ;The length of the message
   dispMsg db 'You have entered: '
   lenDispMsg equ $-dispMsg

section .bss            ;Uninitialized data
   num resb 5

section .text           ;Code Segment
   global _start

_start:                 ;User prompt
   mov eax, 4
   mov ebx, 1
   mov ecx, userMsg
   mov edx, lenUserMsg
   int 80h

   ;Read and store the user input
   mov eax, 3
   mov ebx, 0
   mov ecx, num
   mov edx, 5           ;5 bytes (numeric, 1 for sign) of that information
   int 80h
```

Lời gọi hệ thống Linux: ví dụ


```
;Output the message 'The entered number is: '
mov eax, 4
mov ebx, 1
mov ecx, dispMsg
mov edx, lenDispMsg
int 80h

;Output the number entered
mov eax, 4
mov ebx, 1
mov ecx, num
mov edx, 5
int 80h

; Exit code
mov eax, 1
mov ebx, 0
int 80h
```

Mode địa chỉ

- Các mode địa chỉ
  - Địa chỉ thanh ghi

```
MOV DX, TAX_RATE    ; Register in first operand
MOV COUNT, CX       ; Register in second operand
MOV EAX, EBX        ; Both the operands are in registers
```

  - Địa chỉ tức thời

```
BYTE_VALUE   DB   150     ; A byte value is defined
WORD_VALUE   DW   300     ; A word value is defined
ADD  BYTE_VALUE, 65     ; An immediate operand 65 is added
```

Mode địa chỉ

- Các mode địa chỉ
  - Địa chỉ bộ nhớ

```
MY_TABLE TIMES 10 DW 0  ; Allocates 10 words (2 bytes) each initialized to 0
MOV EBX, [MY_TABLE]     ; Effective Address of MY_TABLE in EBX
MOV [EBX], 110          ; MY_TABLE[0] = 110
ADD EBX, 2              ; EBX = EBX +2
MOV [EBX], 123          ; MY_TABLE[1] = 123
```

Lệnh mov

- Cú pháp: mov dest, src

```
MOV   register, register
MOV   register, immediate
MOV   memory, immediate
MOV   register, memory
MOV   memory, register
```

Type Specifier	Bytes addressed
BYTE	1
WORD	2
DWORD	4
QWORD	8
TBYTE	10

Lệnh mov: ví dụ


```
section .text
   global_start     ;must be declared for linker (ld)
_start:             ;tell linker entry point

   ;writing the name 'Zara Ali'
   mov  edx,9       ;message length
   mov  ecx, name   ;message to write
   mov  ebx,1       ;file descriptor (stdout)
   mov  eax,4       ;system call number (sys_write)
   int  0x80        ;call kernel

   mov  [name],  dword 'Nuha'    ; Changed the name to Nuha Ali

   ;writing the name 'Nuha Ali'
   mov  edx,8       ;message length
   mov  ecx,name    ;message to write
   mov  ebx,1       ;file descriptor (stdout)
   mov  eax,4       ;system call number (sys_write)
   int  0x80        ;call kernel

   mov  eax,1       ;system call number (sys_exit)
   int  0x80        ;call kernel

section .data
name db 'Zara Ali '
```

Biến

- Cú pháp: [var-name] def-directive init-value [,init-value]...

Directive	Purpose	Storage Space
DB	Define Byte	allocates 1 byte
DW	Define Word	allocates 2 bytes
DD	Define Doubleword	allocates 4 bytes
DQ	Define Quadword	allocates 8 bytes
DT	Define Ten Bytes	allocates 10 bytes

```
choice          DB      'y'
number          DW      12345
neg_number      DW      -12345
big_number      DQ      123456789
real_number1    DD      1.234
real_number2    DQ      123.456
```

Biến: ví dụ

- Cú pháp: [var-name] def-directive init-value [,init-value]...

```
section .text
   global _start          ;must be declared for linker (gcc)

_start:                   ;tell linker entry point
   mov  edx,1             ;message length
   mov  ecx,choice        ;message to write
   mov  ebx,1             ;file descriptor (stdout)
   mov  eax,4             ;system call number (sys_write)
   int  0x80              ;call kernel

   mov  eax,1             ;system call number (sys_exit)
   int  0x80              ;call kernel

section .data
choice DB 'y'
```

Biến

- Cú pháp: [var-name] def-directive init-value [,init-value]...

Directive	Purpose
RESB	Reserve a Byte
RESW	Reserve a Word
RESD	Reserve a Doubleword
RESQ	Reserve a Quadword
REST	Reserve a Ten Bytes

Biến: ví dụ

- Cú pháp: [var-name] def-directive init-value [,init-value]...

```
section .text
   global _start          ;must be declared for linker (ld)

_start:                   ;tell linker entry point
   mov  edx,9             ;message length
   mov  ecx, stars        ;message to write
   mov  ebx,1             ;file descriptor (stdout)
   mov  eax,4             ;system call number (sys_write)
   int  0x80              ;call kernel

   mov  eax,1             ;system call number (sys_exit)
   int  0x80              ;call kernel

section .data
stars   times 9 db '*'
```

Hằng

- Cú pháp: [const-name] equ express

```
LENGTH equ 20
WIDTH  equ 10
AREA   equ length * width

TOTAL_STUDENTS equ 50

mov  ecx,  TOTAL_STUDENTS
cmp  eax,  TOTAL_STUDENTS
```

Tính toán số học

- Các phép toán số học

```
inc     dest
dec     dest
add/sub  dest, src          ; dest = dest add/sub src
                            ;  Register to register
                            ;  Memory to register
                            ;  Register to memory
                            ;  Register to constant data
                            ;  Memory to constant data

MUL/IMUL      multiplier
DIV/IDIV      divisor
```

Tính toán số học: ví dụ


```
section .text
   global _start    ;must be declared for using gcc

_start:             ;tell linker entry point
   mov  eax,'3'
   sub  eax, '0'

   mov  ebx, '4'
   sub  ebx, '0'
   add  eax, ebx
   add  eax, '0'

   mov  [sum], eax
   mov  ecx,msg
   mov  edx, len
   mov  ebx,1   ;file descriptor (stdout)
   mov  eax,4   ;system call number (sys_write)
   int  0x80    ;call kernel
```

Tính toán số học: ví dụ


```
    mov   ecx,sum
    mov   edx, 1
    mov   ebx,1    ;file descriptor (stdout)
    mov   eax,4    ;system call number (sys_write)
    int   0x80     ;call kernel

    mov   eax,1    ;system call number (sys_exit)
    int   0x80     ;call kernel

section .data
    msg db "The sum is:", 0xA,0xD
    len equ $ - msg
    segment .bss
    sum resb 1
```

Tính toán số học

- Các phép toán số học

MUL/IMUL multiplier

Tính toán số học: ví dụ


```
section .text
   global _start    ;must be declared for using gcc

_start:             ;tell linker entry point

   mov  al,'3'
   sub  al,'0'

   mov  bl,'2'
   sub  bl,'0'
   mul  bl
   add  al,'0'

   mov  [res],al
   mov  ecx,msg
   mov  edx,len
   mov  ebx,1    ;file descriptor (stdout)
   mov  eax,4    ;system call number (sys_write)
   int  0x80     ;call kernel
```

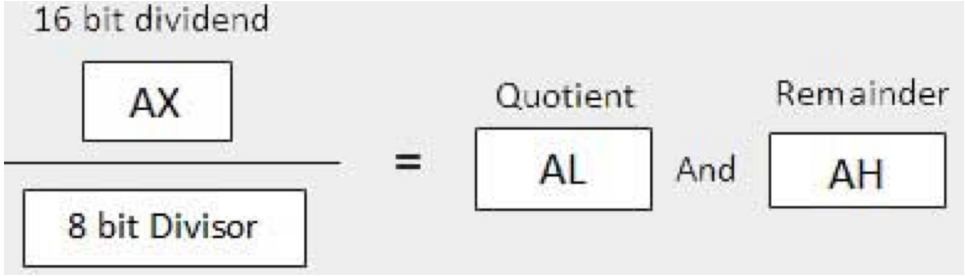
Tính toán số học: ví dụ

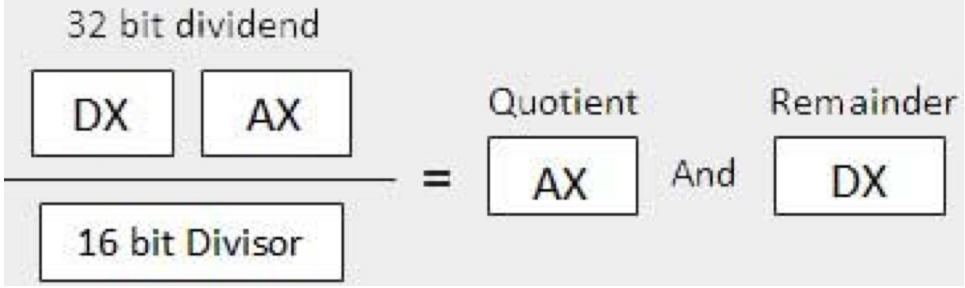

```
    mov   ecx,res
    mov   edx,1
    mov   ebx,1    ;file descriptor (stdout)
    mov   eax,4    ;system call number (sys_write)
    int   0x80     ;call kernel

    mov   eax,1    ;system call number (sys_exit)
    int   0x80     ;call kernel

section .data
msg db "The result is:", 0xA,0xD
len equ $- msg
segment .bss
res resb 1
```

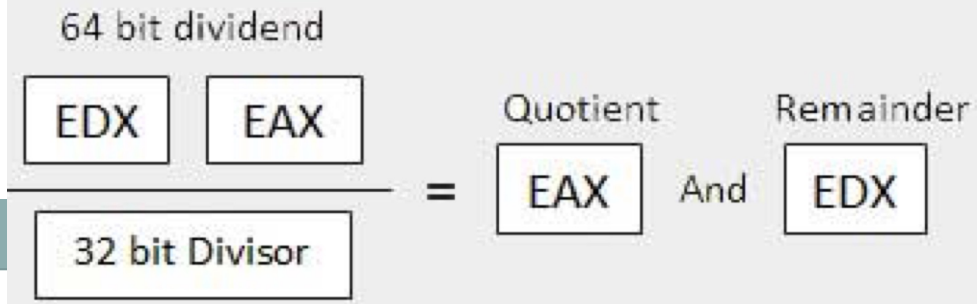
Tính toán số học

- Các phép toán số học

DIV/IDIV divisor







Tính toán số học: ví dụ


```
section .text
   global _start    ;must be declared for using gcc

_start:             ;tell linker entry point
   mov  ax,'8'
   sub  ax, '0'

   mov  bl, '2'
   sub  bl, '0'
   div  bl
   add  ax, '0'

   mov  [res],ax
   mov  ecx,msg
   mov  edx,len
   mov  ebx,1   ;file descriptor (stdout)
   mov  eax,4   ;system call number (sys_write)
   int  0x80    ;call kernel
```

Tính toán số học: ví dụ


```
    mov   ecx,res
    mov   edx,1
    mov   ebx,1    ;file descriptor (stdout)
    mov   eax,4    ;system call number (sys_write)
    int   0x80     ;call kernel

    mov   eax,1    ;system call number (sys_exit)
    int   0x80     ;call kernel

section .data
msg db "The result is:", 0xA,0xD
len equ $- msg
segment .bss
res resb 1
```

Phép toán luận lý

- Các phép toán luận lý

Instruction	Format
AND	AND operand1, operand2
OR	OR operand1, operand2
XOR	XOR operand1, operand2
TEST	TEST operand1, operand2
NOT	NOT operand1

Phép toán luận lý: ví dụ


```
section .text
   global _start                    ;must be declared for using gcc

_start:                             ;tell linker entry point
   mov   ax,   8h                   ;getting 8 in the ax
   and   ax, 1                      ;and ax with 1
   jz    evnn
   mov   eax, 4                     ;system call number (sys_write)
   mov   ebx, 1                     ;file descriptor (stdout)
   mov   ecx, odd_msg               ;message to write
   mov   edx, len2                  ;length of message
   int   0x80                       ;call kernel
   jmp   outprog
```

Phép toán luận lý: ví dụ


```
evnn:

   mov   ah,  09h
   mov   eax, 4              ;system call number (sys_write)
   mov   ebx, 1              ;file descriptor (stdout)
   mov   ecx, even_msg       ;message to write
   mov   edx, len1           ;length of message
   int   0x80                ;call kernel

outprog:

   mov   eax,1               ;system call number (sys_exit)
   int   0x80                ;call kernel

section   .data
even_msg  db  'Even Number!' ;message showing even number
len1  equ  $ - even_msg

odd_msg db  'Odd Number!'    ;message showing odd number
len2  equ  $ - odd_msg
```

Điều kiện

- Các lệnh:

jmp label

j<cond> label

cmp dest, src

Instruction	Description	Flags tested
JE/JZ	Jump Equal or Jump Zero	ZF
JNE/JNZ	Jump not Equal or Jump Not Zero	ZF
JG/JNLE	Jump Greater or Jump Not Less/Equal	OF, SF, ZF
JGE/JNL	Jump Greater/Equal or Jump Not Less	OF, SF
JL/JNGE	Jump Less or Jump Not Greater/Equal	OF, SF
JLE/JNG	Jump Less/Equal or Jump Not Greater	OF, SF, ZF

Điều kiện

- Các lệnh:

jmp label

j<cond> label

cmp dest, src

Instruction	Description	Flags tested
JE/JZ	Jump Equal or Jump Zero	ZF
JNE/JNZ	Jump not Equal or Jump Not Zero	ZF
JA/JNBE	Jump Above or Jump Not Below/Equal	CF, ZF
JAE/JNB	Jump Above/Equal or Jump Not Below	CF
JB/JNAE	Jump Below or Jump Not Above/Equal	CF
JBE/JNA	Jump Below/Equal or Jump Not Above	AF, CF

Điều kiện

- Các lệnh:

jmp label

j<cond> label

cmp dest, src

Instruction	Description	Flags tested
JXCZ	Jump if CX is Zero	none
JC	Jump If Carry	CF
JNC	Jump If No Carry	CF
JO	Jump If Overflow	OF
JNO	Jump If No Overflow	OF
JP/JPE	Jump Parity or Jump Parity Even	PF
JNP/JPO	Jump No Parity or Jump Parity Odd	PF
JS	Jump Sign (negative value)	SF
JNS	Jump No Sign (positive value)	SF

Điều kiện: ví dụ


```
section .text
   global _start          ;must be declared for using gcc

_start:                   ;tell linker entry point
   mov   ecx, [num1]
   cmp   ecx, [num2]
   jg    check_third_num
   mov   ecx, [num2]

    check_third_num:

   cmp   ecx, [num3]
   jg    _exit
   mov   ecx, [num3]

    _exit:

   mov   [largest], ecx
   mov   ecx,msg
   mov   edx, len
   mov   ebx,1  ;file descriptor (stdout)
   mov   eax,4  ;system call number (sys_write)
   int   0x80   ;call kernel
```

Điều kiện: ví dụ


```
    mov   ecx,largest
    mov   edx, 2
    mov   ebx,1  ;file descriptor (stdout)
    mov   eax,4  ;system call number (sys_write)
    int   0x80   ;call kernel

    mov   eax, 1
    int   80h

section .data

    msg db "The largest digit is: ", 0xA,0xD
    len equ $- msg
    num1 dd '47'
    num2 dd '22'
    num3 dd '31'

segment .bss
    largest resb 2
```

Vòng lặp


```
MOV      CL, 10
L1:
<LOOP-BODY>
DEC      CL
JNZ      L1
```

```
mov ECX,10
l1:
<loop body>
loop l1
```

Vòng lặp: ví dụ


```
section .text
   global _start        ;must be declared for using gcc

_start:                 ;tell linker entry point
   mov ecx,10
   mov eax, '1'

l1:
   mov [num], eax
   mov eax, 4
   mov ebx, 1
   push ecx

   mov ecx, num
   mov edx, 1
   int 0x80
```

Vòng lặp: ví dụ


```
    mov eax, [num]
    sub eax, '0'
    inc eax
    add eax, '0'
    pop ecx
    loop l1

    mov eax,1               ;system call number (sys_exit)
    int 0x80                ;call kernel
section .bss
num resb 1
```

Mảng


```
NUMBERS DW  34,  45,  56,  67,  75, 89
INVENTORY    DW   0
             DW   0
             DW   0
             DW   0
             DW   0
             DW   0
             DW   0
             DW   0
INVENTORY    DW  0, 0 , 0 , 0 , 0 , 0 , 0 , 0
INVENTORY TIMES 8 DW 0
```

Mảng: ví dụ


```
section .text
   global _start    ;must be declared for linker (ld)

_start:

   mov  eax,3       ;number bytes to be summed
   mov  ebx,0       ;EBX will store the sum
   mov  ecx, x      ;ECX will point to the current element to be
       summed

top:  add  ebx, [ecx]

   add  ecx,1       ;move pointer to next element
   dec  eax         ;decrement counter
   jnz  top         ;if counter not 0, then loop again

done:

   add   ebx, '0'
   mov  [sum], ebx ;done, store result in "sum"
```

Mảng: ví dụ


```
display:

   mov  edx,1       ;message length
   mov  ecx, sum    ;message to write
   mov  ebx, 1      ;file descriptor (stdout)
   mov  eax, 4      ;system call number (sys_write)
   int  0x80        ;call kernel

   mov  eax, 1      ;system call number (sys_exit)
   int  0x80        ;call kernel

section .data
global x
x:
   db  2
   db  4
   db  3

sum:
   db  0
```

Thủ tục


```
proc_name:
    procedure body
    ...
    ret

CALL proc_name
```

Thủ tục: ví dụ


```
section .text
   global _start        ;must be declared for using gcc

_start:                 ;tell linker entry point
   mov  ecx,'4'
   sub  ecx, '0'

   mov  edx, '5'
   sub  edx, '0'

   call sum          ;call sum procedure
   mov  [res], eax
   mov  ecx, msg
   mov  edx, len
   mov  ebx,1           ;file descriptor (stdout)
   mov  eax,4           ;system call number (sys_write)
   int  0x80            ;call kernel

   mov  ecx, res
   mov  edx, 1
   mov  ebx, 1          ;file descriptor (stdout)
   mov  eax, 4          ;system call number (sys_write)
   int  0x80            ;call kernel
```

Thủ tục: ví dụ


```
    mov  eax,1              ;system call number (sys_exit)
    int  0x80               ;call kernel
sum:
    mov     eax, ecx
    add     eax, edx
    add     eax, '0'
    ret

section .data
msg db "The sum is:", 0xA,0xD
len equ $- msg

segment .bss
res resb 1
```

Thủ tục: ví dụ


```
section .text
    global _start          ;must be declared for using gcc
_start:                    ;tell linker entry point
    call    display
    mov  eax,1             ;system call number (sys_exit)
    int  0x80              ;call kernel
display:
    mov    ecx, 256
next:
    push    ecx
    mov     eax, 4
    mov     ebx, 1
    mov     ecx, achar
    mov     edx, 1
    int     80h

    pop     ecx
    mov  dx, [achar]
    cmp  byte [achar], 0dh
    inc  byte [achar]
    loop    next
    ret
```

```
section .data
achar db '0'
```

Thủ tục: ví dụ


```
section .text
   global _start          ;must be declared for using gcc

_start:                   ;tell linker entry point

   mov bx, 3              ;for calculating factorial 3
   call  proc_fact
   add   ax, 30h
   mov  [fact], ax

   mov    edx,len         ;message length
   mov    ecx,msg         ;message to write
   mov    ebx,1           ;file descriptor (stdout)
   mov    eax,4           ;system call number (sys_write)
   int    0x80            ;call kernel

   mov   edx,1             ;message length
   mov    ecx,fact        ;message to write
   mov    ebx,1           ;file descriptor (stdout)
   mov    eax,4           ;system call number (sys_write)
   int    0x80            ;call kernel
```

Thủ tục: ví dụ


```
    mov     eax,1           ;system call number (sys_exit)
    int     0x80            ;call kernel

proc_fact:
    cmp     bl, 1
    jg      do_calculation
    mov     ax, 1
    ret

do_calculation:
    dec     bl
    call    proc_fact
    inc     bl
    mul     bl          ;ax = al * bl
    ret

section .data
msg db 'Factorial 3 is:',0xa
len equ $ - msg

section .bss
fact resb 1
```

Macro


```
%macro macro_name  number_of_params
<macro body>
%endmacro
```

Macro: ví dụ


```
; A macro with two parameters
; Implements the write system call
   %macro write_string 2
      mov   eax, 4
      mov   ebx, 1
      mov   ecx, %1
      mov   edx, %2
      int   80h
   %endmacro
section .text
   global _start            ;must be declared for using gcc
_start:                     ;tell linker entry point
   write_string msg1, len1
   write_string msg2, len2
   write_string msg3, len3
   mov eax,1                ;system call number (sys_exit)
   int 0x80                 ;call kernel
section .data
msg1 db 'Hello, programmers!',0xA,0xD
len1 equ $ - msg1
msg2 db 'Welcome to the world of,', 0xA,0xD
len2 equ $- msg2
msg3 db 'Linux assembly programming! '
len3 equ $- msg3
```

Quản lý tập tin

- 3 tập tin thiết bị số hiệu 0, 1, 2:
- Tập tin: thẻ file, con trỏ file

Standard input (stdin),

Standard output (stdout), and

Standard error (stderr).

%eax	Name	%ebx	%ecx	%edx
2	sys_fork	struct pt_regs	-	-
3	sys_read	unsigned int	char *	size_t
4	sys_write	unsigned int	const char *	size_t
5	sys_open	const char *	int	int
6	sys_close	unsigned int	-	-
8	sys_creat	const char *	int	-
19	sys_lseek	unsigned int	off_t	unsigned int

Quản lý tập tin

- Tạo và mở tập tin
  - system call sys_creat() number 8 => EAX
  - filename => EBX
  - file permissions => ECX
  - file descriptor/error <= EAX
- Mở tập tin có sẵn
  - sys_open() number 5 => EAX
  - filename => EBX register
  - access mode (read-only:0, write-only:1, read-write:2) => ECX
  - file permissions => EDX
  - file descriptor/error <= EAX

Quản lý tập tin

- Đọc tập tin
  - system call sys_read() number 3 => EAX
  - file descriptor => EBX
  - pointer to the input buffer => ECX
  - buffer size, i.e., the number of bytes to read => EDX
  - number of bytes read/error <= EAX
- Ghi tập tin
  - system call sys_write() number 4 => EAX
  - file descriptor => EBX
  - pointer to the output buffer => ECX
  - buffer size, i.e., the number of bytes to write => EDX
  - actual number of bytes written/error <= EAX

Quản lý tập tin

- Đóng tập tin
  - system call sys_close() number 6 => EAX
  - file descriptor => EBX
  - error <= EAX
- Cập nhật tập tin
  - system call sys_lseek () number 19 => EAX
  - file descriptor => EBX
  - offset value => ECX
  - reference position for the offset (0: beginning of file, 1: current position, 2: End of file) => EDX

Quản lý tập tin: ví dụ


```
section .text
   global _start          ;must be declared for using gcc

_start:                   ;tell linker entry point
   ;create the file
   mov  eax, 8
   mov  ebx, file_name
   mov  ecx, 0777         ;read, write and execute by all
   int  0x80              ;call kernel

   mov [fd_out], eax

   ; write into the file
   mov  edx,len           ;number of bytes
   mov  ecx, msg          ;message to write
   mov  ebx, [fd_out]     ;file descriptor
   mov  eax,4             ;system call number (sys_write)
   int  0x80              ;call kernel

   ; close the file
   mov eax, 6
   mov ebx, [fd_out]
```

Quản lý tập tin: ví dụ


```
; write the message indicating end of file write
mov eax, 4
mov ebx, 1
mov ecx, msg_done
mov edx, len_done
int  0x80

;open the file for reading
mov eax, 5
mov ebx, file_name
mov ecx, 0              ;for read only access
mov edx, 0777           ;read, write and execute by all
int  0x80

mov  [fd_in], eax

;read from file
mov eax, 3
mov ebx, [fd_in]
mov ecx, info
mov edx, 26
int 0x80
```

Quản lý tập tin: ví dụ


```
; close the file
mov eax, 6
mov ebx, [fd_in]
int  0x80
; print the info
mov eax, 4
mov ebx, 1
mov ecx, info
mov edx, 26
int 0x80
mov  eax,1              ;system call number (sys_exit)
int  0x80               ;call kernel
section .data
file_name db 'myfile.txt'
msg db 'Welcome to Tutorials Point'
len equ  $-msg
msg_done db 'Written to file', 0xA,0xD
len_done equ $-msg_done
```

Quản lý tập tin: ví dụ


```
section .bss
fd_out resb 1
fd_in  resb 1
info resb  26
```