**BÁO CÁO THỰC HÀNH**

# LẬP TRÌNH HỆ THỐNG

## Tên bài Thực hành: Lab 03 - Lớp: NT209.P12.ANTT

Giáo viên hướng dẫn: Đỗ Thị Thu Hiền

| Họ và tên Sinh viên | MSSV |
|---|---|
| Nguyễn Trần Minh Khôi | 23520780 |
| Lê Đăng Khôi | 23520766 |
| Vương Thành Đạt | 23520281 |

### C.1 Chương trình kiểm tra tính tăng dần của các chữ số trong số có 4 chữ số

**Input:** Một số nguyên a có 4 chữ số ($1000 \leq a \leq 9999$)

**Output:** Xuất ra màn hình nhận định: "**Tang dan**" nếu các chữ số trong a tăng dần, ngược lại xuất ra "**Khong tang dan**"

Hình ảnh:

```
1 .section .data
2 tang: .string "Tang dan"
3 ktang: .string "Khong tang dan"
4
5 .section .bss
6 .lcomm input, 4
7 .lcomm outputT, 9
8 .lcomm outputK, 15
9
10 .section .text
11 .globl _start
12
13 _start:
14     # Read 4 bytes from stdin into input
15     movl $3, %eax
16     movl $0, %ebx
17     movl $input, %ecx
18     movl $4, %edx
19     int $0×80
20
21     movl $input, %esi
22
23     # Convert the first character to a number
24     movzbl 0(%esi), %eax
25     sub $'0', %al
26     mov %al, %bh
```

```
28     # Convert the second character to a number and compare
29     movzbl 1(%esi), %eax
30     sub $'0', %al
31     mov %al, %bl
32     cmp %bh, %bl
33     jle .NOT_INCREASING
34     mov %bl, %bh
35
36     # Convert the third character to a number and compare
37     movzbl 2(%esi), %eax
38     sub $'0', %al
39     mov %al, %bl
40     cmp %bh, %bl
41     jle .NOT_INCREASING
42     mov %bl, %bh
43
44     # Convert the fourth character to a number and compare
45     movzbl 3(%esi), %eax
46     sub $'0', %al
47     mov %al, %bl
48     cmp %bh, %bl
49     jle .NOT_INCREASING
```

```
51      # Print "Tang dan"
52      movl $4, %eax
53      movl $1, %ebx
54      movl $tang, %ecx
55      movl $9, %edx
56      int $0×80
57      jmp .END
58
59  .NOT_INCREASING:
60      # If not increasing, print "Khong tang dan"
61      movl $4, %eax
62      movl $1, %ebx
63      movl $ktang, %ecx
64      movl $15, %edx
65      int $0×80
66
67  .END:
68      movl $1, %eax
69      int $0×80
70
```

Giải thích:

Trong section .data:

+ Khai báo hai dòng thông báo “Tăng dần”, “Không tăng dần” để in ra màn hình.

Trong section .bss:

+ Khai báo vùng nhớ mang input có 4 byte, dùng để chứa số nhập vào và xét xem tăng dần hay không tăng dần.

+ Vùng nhớ outputT có 9 byte, dùng để chứa dòng thông báo “Tăng dần”

+ Vùng nhớ outputK có 15 byte, dùng để chứa dòng thông báo “Không tăng dần”

Trong _start:

```
# Read 4 bytes from stdin into input
movl $3, %eax
movl $0, %ebx
movl $input, %ecx
movl $4, %edx
int $0x80
```

movl $3, %eax**:** Đặt giá trị 3 vào thanh ghi EAX để chỉ định hàm gọi hệ thống đọc (syscall read).
movl $0, %ebx**:** Đặt giá trị 0 vào thanh ghi EBX để chỉ định đầu vào chuẩn (stdin) là nguồn đọc.
movl $input, %ecx**:** Đặt địa chỉ của biến input vào thanh ghi ECX để chỉ định bộ đệm nơi lưu dữ liệu đọc vào.
movl $4, %edx**:** Đặt giá trị 4 vào thanh ghi EDX để chỉ định số byte muốn đọc.
int $0x80**:** Gọi hệ thống để thực hiện hàm đọc dựa trên các thanh ghi đã thiết lập trước đó.

Mục đích: Chức năng chính của những dòng mã này là đọc vào 4 byte dữ liệu từ đầu vào chuẩn (stdin) và lưu dữ liệu đó vào bộ đệm input.

```
movl $input, %esi
# Convert the first character to a number
```

```
movzbl 0(%esi), %eax
sub $'0', %al
mov %al, %bh
```

movl $input, %esi: Chuyển giá trị của vùng nhớ input vào trong thanh ghi ESI, phục vụ cho việc tính toán sau này.
movzbl 0(%esi), %eax: Chuyển đổi byte đầu tiên trong thanh ghi %ESI lên cho thanh ghi %eax
sub $'0', %al: Trừ đi giá trị ASCII của ký tự '0' từ thanh ghi AL. Điều này chuyển đổi ký tự thành giá trị số tương ứng
mov %al, %bh: Di chuyển giá trị đã chuyển đổi trong AL vào thanh ghi BH.

Mục đích: Lấy ra kí tự đầu tiên trong số input nhập vào, sau đó chuyển nó thành số và lưu lại vào thanh ghi BH.

```
# Convert the second character to a number and compare
movzbl 1(%esi), %eax
sub $'0', %al
mov %al, %bl
cmp %bh, %bl
jle .NOT_INCREASING
mov %bl, %bh
```

movzbl 1(%esi), %eax: Chuyển đổi byte thứ hai (ký tự thứ hai) tại địa chỉ mà ESI trỏ tới thành một số và lưu vào EAX.

sub $'0', %al: Trừ đi giá trị ASCII của ký tự '0' từ thanh ghi AL. Điều này chuyển đổi ký tự thành giá trị số tương ứng
mov %al, %bl: Di chuyển giá trị đã chuyển đổi trong AL vào thanh ghi BH.

cmp %bh, %bl: So sánh giá trị bên trong hai thanh ghi %bh, %bl

jle .NOT_INCREASING: Nhảy đến nhãn NOT_INCREASING nếu giá trị trong %bl nhỏ hơn %bh

mov %bl, %bh: Di chuyển giá trị %bh vào trong %bl để phục vụ cho việc tính toán và so sánh với số tiếp theo.

Tiếp tục thực hiện việc đưa byte thứ ba và thứ tư của input vào trong từng thanh ghi và so sánh các kết quả với nhau

Nếu tất cả đều theo quy luật số sau lớn hơn số trước thì

```
# Print "Tang dan"
 movl $4, %eax
 movl $1, %ebx
 movl $tang, %ecx
 movl $9, %edx
 int $0x80
 jmp .END
```

In ra dòng thông báo “Tăng dần” bằng system call và nhảy đến nhãn .END để kết thúc chương trình

Nếu một trong những byte của 4 byte input (4 ký tự số trong 4 số input) không theo quy luật số sau lớn hơn số trước thì nhảy đến nhãn .NOT_INCREASING:

```
.NOT_INCREASING:
   # If not increasing, print "Khong tang dan"
   movl $4, %eax
   movl $1, %ebx
   movl $ktang, %ecx
   movl $15, %edx
   int $0x80
```

Thực hiện việc in ra dòng thông báo "Không tăng dần" bằng system call và kết thúc chương trình bằng nhãn .END

```
.END:
   movl $1, %eax
   int $0x80
```

Gọi system call để kết thúc chương trình.

Kết quả:

```
┌──(kali㉿kali)-[~/Documents/LTHT/LAB_03]
└─$ as c31.s -o c31.o

┌──(kali㉿kali)-[~/Documents/LTHT/LAB_03]
└─$ ld -o c31 c31.o

┌──(kali㉿kali)-[~/Documents/LTHT/LAB_03]
└─$ ./c31
1234
Tang dan

┌──(kali㉿kali)-[~/Documents/LTHT/LAB_03]
└─$

┌──(kali㉿kali)-[~/Documents/LTHT/LAB_03]
└─$ ./c31
2234
Khong tang dan
```

## C.2 Chuẩn hóa chuỗi ký tự có 10 ký tự

**Input:** Nhập 1 chuỗi có 10 ký tự (hoa hoặc thường)

**Output:** Xuất ra chuỗi đã chuẩn hóa: Chữ cái đầu mỗi từ viết hoa, các chữ cái còn lại viết thường.

**Nâng cao:** Xử lý chuỗi có độ dài không quá 255 ký tự, trong đó có thể có nhiều dấu cách bị lặp. Trong chuỗi có ký tự đặc biệt và số thì giữ nguyên.

- Hình ảnh:

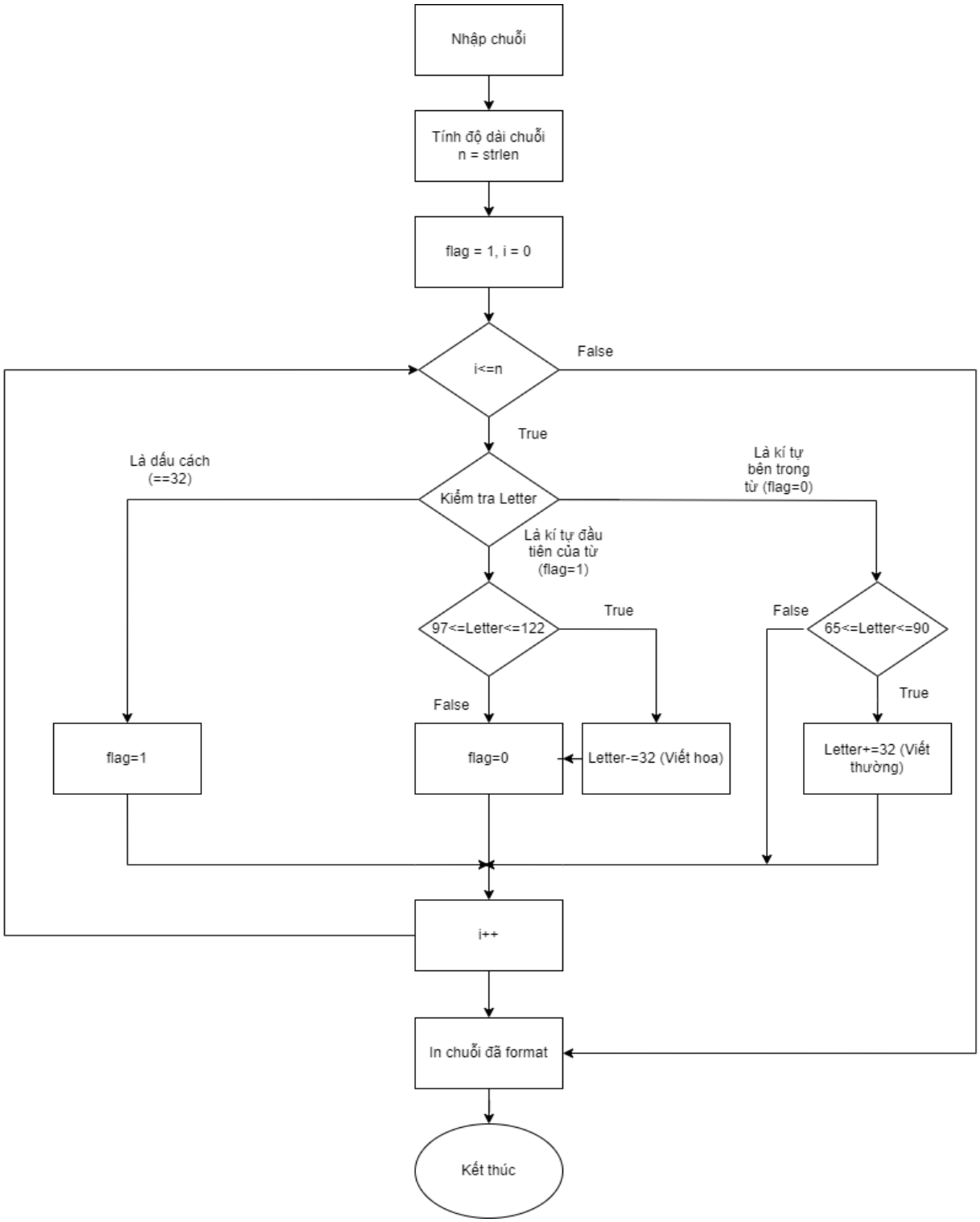
```
.section .data
   prompt: .asciz "Enter a string (<255 chars): "
.section .bss
   .lcomm flag, 1
   .lcomm string, 255
   .lcomm ketqu, 255
```

```
    .lcomm n, 4
.section .text
    .globl _start
_start:
    movl $4, %eax
    movl $1, %ebx
    movl $prompt, %ecx
    movl $28, %edx
    int $0x80

    //Nhap chuoi
    movl $3, %eax
    movl $0, %ebx
    movl $string, %ecx
    movl $255, %edx
    int $0x80

    movl $0, %ecx
    jmp .test
.for:
    addl $1, %ecx
.test:
    movl $0, %ebx
    addl $string, %ebx
    addl %ecx, %ebx
    movb (%ebx), %ah
    cmpb $10, %ah
    je .exit_for
    jne .for
.exit_for:

    .while:
    cmpl n, %ecx
    jge .exit_while

    movl $0, %ebx
    addl $string, %ebx
    add %ecx, %ebx

    movb (%ebx), %ah

    cmpb $32, %ah
    je .space

    cmpb $1, flag
    je .first_letter_of_word
    jne .in_word_letter

.space:
    movb $1, flag
    jmp .update

.first_letter_of_word:
```

```
    cmpb $97, %ah
    jl .change.flag
    cmpb $122, %ah
    jg .change.flag
    subb $32, %ah
    movb %ah, (%ebx)
    movb $0, flag
    jmp .update

.change.flag:
    movb $0, flag
    jmp .update

.in_word_letter:
    cmpb $65, %ah
    jl .update
    cmpb $90, %ah
    jg .update
    addb $32, %ah
    movb %ah, (%ebx)
    jmp .update

.update:
    addl $1, %ecx
    jmp .while

.exit_while:
    movl $4, %eax
    movl $1, %ebx
    movl $string, %ecx
    movl n, %edx
    int $0x80

    movl $1, %eax
    int $0x80
```

Giải thích:

Lưu đồ thuật toán:




- Giải thích một số hàm:
  - Hàm tính độ dài chuỗi:

```
char Chuoi[255]
        int n=0;
        for (int i=0;i<=255;i++)
        if(a[i]!=”/n”) n++;
        else break;
```

  - Hàm format chuỗi:

```
int i=0;
bool flag=1;
 while(i<=n)
```

```
        {
if(Chuoi[i]==” “) flag=1;
else if(flag==1&&a<=Chuoi[i]<=z) toupper(Chuoi[i]);
      else if(flag=0&&A<=Chuoi[i]<=Z) tolower(Chuoi[i]);
i++
}
```

Kết quả:

```
Enter a string (<255 chars):hello attn
Hello Attn
```

```
Enter a string (<255 chars):ClAss aNTt
Class Antt
```

```
Enter a string (<255 chars):heLLo cLAss
Hello Class
```

```
Enter a string (<255 chars):l@vE Uit 2024
L@ve Uit 2024
```

### C.3 Chương trình đếm số chia hết cho 4 trong 5 số có 1 chữ số

**Input:** Nhập 5 số nguyên có 1 chữ số ($0 \leq x \leq 9$)

**Output:** Xuất ra màn hình số lượng số chia hết cho 4.

Hình ảnh:

```
.section .data
   input_string: .string "Enter a number (1 - digit): "
   input_string_length = .-input_string
   output_string: .string "Count 4x: "
   output_string_length = .- output_string

.section .bss
   .lcomm input  2
   .lcomm count  2
   .lcomm output 2

.section .text
   .globl _start

_start:
   # Khoi tao count =0
   movl $0, count
   # Khoi tao chi so i = %esi
   movl $0, %esi
# Vong lap nhap input
loop_start:
   cmpl $5, %esi
   jge print_result
```

```
    # In thong bao nhap
    movl $4, %eax
    movl $1, %ebx
    movl $input_string, %ecx
    movl $input_string_length, %edx
    int $0x80
    # Nhap input
    movl $3, %eax
    movl $0, %ebx
    movl $input, %ecx
    movl $2, %edx
    int $0x80
    # Chuyen input thanh so
    movb input, %al
    subb $'0', %al
    movzx %al, %eax
    # Kiem tra chia het cho 4
    movl $4, %ebx
    xorl %edx, %edx
    divl %ebx
    cmpl $0, %edx
    jne continue_loop
    addl $1, count
continue_loop:
    # Tang i len 1 don vi va tiep tuc vong lap
    addl $1, %esi
    jmp loop_start
print_result:
    # Chuyen count sang %eax
    movl count, %eax
    # Chuyen count thanh ky tu
    addl $'0', %eax
    movb %al, output
    movb $0x0A, output+1
    # In thong bao "Count 4x: "
    movl $4, %eax
    movl $1, %ebx
    movl $output_string, %ecx
    movl $output_string_length, %edx
    int $0x80
    # In ket qua
    movl $4, %eax
    movl $1, %ebx
    movl $output, %ecx
    movl $2, %edx
    int $0x80
    # Thoat chuong trinh
    movl $1, %eax
    xorl %ebx, %ebx
    int $0x80
```

Giải thích:

- Phần khai báo dữ liệu .section .data:
  - Khởi tạo chuỗi input_string và output_string có sẵn dữ liệu và khai báo độ dài của chuỗi.
  - Với input_string sẽ là chuỗi "Enter a number (1 - digit): " và output_string là chuỗi "Count 4x: ".
- Phần khai báo vùng nhớ .section .bss:
  - Dành bộ nhớ chưa khởi tạo các biến input, count và output với độ dài lần lượt là 2,2,2 bytes:
    - Dữ liệu của input sẽ tương ứng với giá trị sau khi hiển thị input_string.
    - Dữ liệu của count sẽ như một biến đếm lưu giá trị để gọi lại hàm input_string.
    - Dữ liệu của output sẽ tương ứng với giá trị cuối cùng sau khi hiển thị output_string.
- Phần mã lệnh .section .text:
  - Khai báo hàm _start để bắt đầu các mã lệnh trong nó. Hàm _start như một biến toàn cục của cả chương trình.
- Hàm _start:
  - Phần đầu là khởi tạo giá trị của count = 0 và gán 0 thành giá trị của thanh ghi chỉ số esi
- Hàm .loop_start: Vòng lặp nhập dữ liệu
  - In thông báo nhập:
    - Thực hiện so sánh giá trị của thanh ghi esi với 5 bằng lệnh cmpl. Nếu thanh ghi esi có giá trị lớn hơn hoặc bằng 5 thì sẽ nhảy đến hàm print_result.
    - Sau đó, thực hiện thao tác in thông báo nhập với chuỗi là input_string có độ dài input_string_length. Tiếp đến, thực hiện thao tác nhập giá trị để lưu vào biến input có độ dài 2 bytes
    - Xử lý dữ liệu nhập: giá trị input được nhập vào là kiểu chuỗi, ta phải chuyển input thành kiểu số.
      - **movb input, %al**: Tải byte đầu tiên của dữ liệu vào %al.
      - **subb $'0', %al**: Trừ đi giá trị ASCII của '0' để chuyển từ ký tự sang số.
      - **movzx %al, %eax**: Mở rộng số không dấu từ %al sang %eax.
    - Kiểm tra chia hết cho 4:
      - **movl $4, %ebx**: Tải 4 vào %ebx.
      - **xorl %edx, %edx**: Xóa giá trị của %edx (để chuẩn bị cho phép chia).
      - **divl %ebx**: Chia %eax cho %ebx, kết quả nằm trong %eax, phần dư nằm trong %edx.
      - **cmpl $0, %edx**: So sánh phần dư với 0.

- **jne continue_loop**: Nếu không bằng (không chia hết), nhảy đến continue_loop.
- **addl $1, count**: Tăng count nếu chia hết.

- Hàm tiếp tục vòng lặp .continue_loop:
  - Tăng i và tiếp tục vòng lặp:
    - **addl $1, %esi**: Tăng chỉ số i lên 1.
    - **jmp loop_start**: Nhảy về đầu vòng lặp.
- Hàm in kết quả .print_result:
  - Thực hiện chuyển giá trị count vào thanh ghi eax sau đó chuyển về kiểu chuỗi và lưu vào output
    - **print_result:**: Đánh dấu nơi để in kết quả.
    - **movl count, %eax**: Tải count vào %eax.
    - **addl $'0', %eax**: Chuyển đổi số về dạng ký tự ASCII.
    - **movb %al, output**: Lưu ký tự ASCII vào output.
    - **movb $0x0A, output+1**: Lưu ký tự xuống dòng vào vị trí tiếp theo của output.
  - In kết quả ra màn hình:
  - **In Kết Quả**: Hai khối lệnh này in ra nhãn và kết quả.
  - **Gọi Hệ Thống Để Ghi**: Lặp lại tương tự như trước để in chuỗi và kết quả.
- Thoát chương trình:
    - **movl $1, %eax**: Chỉ định gọi hệ thống sys_exit.
    - **xorl %ebx, %ebx**: Đặt mã thoát là 0.
    - **int $0x80**: Kích hoạt kernel để thoát.
- Phép chia **divl** trong assembly x86 hoạt động khá đặc biệt:
- Số bị chia là giá trị 64-bit được tạo bởi cặp thanh ghi EDX:EAX
  - EDX chứa 32 bit cao
  - EAX chứa 32 bit thấp
- Số chia là toán hạng trong lệnh (ở đây là giá trị trong %ebx - số 4)
- Sau phép chia:
  - Thương số được lưu trong EAX
  - Số dư được lưu trong EDX
  - Ví dụ cụ thể:
    - Giả sử EAX = 12 (số cần kiểm tra)
    - EDX = 0 (xorl %edx, %edx đã set về 0)
    - EBX = 4 (số chia)
    - Khi thực hiện divl %ebx:
    - 12 ÷ 4 = 3 dư 0
    - EAX sẽ chứa 3 (thương số)
    - EDX sẽ chứa 0 (số dư)

Kết quả:

```
Enter a number (1 - digit): 3
Enter a number (1 - digit): 8
Enter a number (1 - digit): 5
Enter a number (1 - digit): 1
Enter a number (1 - digit): 7
Count 4x: 1
```

```
Enter a number (1 - digit): 0
Enter a number (1 - digit): 8
Enter a number (1 - digit): 3
Enter a number (1 - digit): 2
Enter a number (1 - digit): 4
Count 4x: 3
```

### C.4 Kiểm tra tình trạng sinh viên dựa trên năm sinh

**Input:** Nhập năm sinh gồm 4 chữ số. Giả sử một SV bắt đầu học ở UIT từ 18 tuổi và theo kế hoạch sẽ tốt nghiệp lúc 22 tuổi.

**Output:** Xuất ra màn hình nhận định:

+ "Chua vao UIT" nếu tuổi < 18
+ "Dang hoc tai UIT" nếu tuổi từ 18 - 22
+ "Da tot nghiep" nếu tuổi > 22

**Nâng cao:** In thêm số tuổi, đồng thời với 3 trường hợp trên:

+ Nếu tuổi < 18, in ra năm dự kiến vào UIT.
+ Nếu tuổi 18 – 22, in ra năm dự kiến tốt nghiệp.
+ Nếu tuổi > 22, in ra năm đã tốt nghiệp.

Hình ảnh:

```
.section .data
  input_string: .string "Nhap nam sinh gom 4 chu so: "
  input_string_length = .-input_string
  Dang_hoc_tai_UIT: .string "Dang hoc tai UIT\n"
  Dang_hoc_tai_UIT_length = .-Dang_hoc_tai_UIT
  Chua_vao_UIT: .string "Chua vao UIT\n"
  Chua_vao_UIT_length = .-Chua_vao_UIT
  Da_tot_nghiep: .string "Da tot nghiep\n"
  Da_tot_nghiep_length = .-Da_tot_nghiep
  Tuoi: .string "Tuoi: "
  Tuoi_length = .-Tuoi
  Du_kien_tot_nghiep: .string "\nDu kien tot nghiep: "
  Du_kien_tot_nghiep_length = .-Du_kien_tot_nghiep
  Du_kien_vao_UIT: .string "\nDu kien vao UIT: "
  Du_kien_vao_UIT_length = .-Du_kien_vao_UIT
  Nam_da_tot_nghiep: .string "\nNam da tot nghiep: "
  Nam_da_tot_nghiep_length = .-Nam_da_tot_nghiep
  newline: .string "\n"
```

```
    space: .string " "
    buffer: .space 10
    num_buffer: .space 10

.section .bss
    .lcomm NamSinh, 4
    .lcomm TuoiHT, 4
    .lcomm KetQua, 4
    .lcomm TempNum, 4

.section .text
.globl _start
_start:
    # In thong bao nhap
    movl $4, %eax
    movl $1, %ebx
    movl $input_string, %ecx
    movl $input_string_length, %edx
    int $0x80

    # Nhap nam sinh
    movl $3, %eax
    movl $0, %ebx
    movl $buffer, %ecx
    movl $5, %edx    # 4 chu so + newline
    int $0x80

    # Chuyen chuoi thanh so
    xorl %eax, %eax     # Xoa eax
    movl $buffer, %esi   # Dia chi buffer vao esi
    movl $0, %ecx       # Reset bien dem
    movl $10, %ebx
    jmp .test
.for:
    mull %ebx
    addb (%esi), %al
    subb $48, %al
    incl %esi
    incl %ecx
.test:
    cmpl $4, %ecx
    jge .exit_for
    jmp .for
.exit_for:
    movl %eax, NamSinh

    # Tinh tuoi
    movl $2024, %edx    # Nam hien tai
    subl NamSinh, %edx
    movl %edx, TuoiHT

    # So sanh tuoi
```

```
    movl TuoiHT, %eax
    cmpl $18, %eax      # So sanh voi 18
    jl chua_vao_uit     # Neu < 18, nhay toi chua_vao_uit
    cmpl $22, %eax      # So sanh voi 22
    jle dang_hoc        # Neu <= 22, nhay toi dang_hoc
    jmp da_tot_nghiep   # Con lai la da tot nghiep

chua_vao_uit:
    # In "Chua vao UIT"
    movl $4, %eax
    movl $1, %ebx
    movl $Chua_vao_UIT, %ecx
    movl $Chua_vao_UIT_length, %edx
    int $0x80

    # In "Tuoi: "
    movl $4, %eax
    movl $1, %ebx
    movl $Tuoi, %ecx
    movl $Tuoi_length, %edx
    int $0x80

    # Xu ly in tuoi
    movl TuoiHT, %eax
    movl $10, %ebx
    movl $0, %edx
    divl %ebx           # Lay chu so hang chuc
    addl $48, %eax      # Chuyen thanh ky tu
    movb %al, num_buffer
    addl $48, %edx      # Chuyen phan du thanh ky tu
    movb %dl, num_buffer + 1

    # In tuoi
    movl $4, %eax
    movl $1, %ebx
    movl $num_buffer, %ecx
    movl $2, %edx       # In 2 ky tu
    int $0x80

    # In thong bao du kien vao UIT
    movl $4, %eax
    movl $1, %ebx
    movl $Du_kien_vao_UIT, %ecx
    movl $Du_kien_vao_UIT_length, %edx
    int $0x80

    # Tinh nam du kien vao UIT
    movl NamSinh, %eax
    addl $18, %eax
    movl %eax, KetQua
    jmp print_ketqua
```

```
dang_hoc:
    # In "Dang hoc tai UIT"
    movl $4, %eax
    movl $1, %ebx
    movl $Dang_hoc_tai_UIT, %ecx
    movl $Dang_hoc_tai_UIT_length, %edx
    int $0x80

    # In "Tuoi: "
    movl $4, %eax
    movl $1, %ebx
    movl $Tuoi, %ecx
    movl $Tuoi_length, %edx
    int $0x80

    # Xu ly in tuoi
    movl TuoiHT, %eax
    movl $10, %ebx
    movl $0, %edx
    divl %ebx          # Lay chu so hang chuc
    addl $48, %eax     # Chuyen thanh ky tu
    movb %al, num_buffer
    addl $48, %edx     # Chuyen phan du thanh ky tu
    movb %dl, num_buffer + 1

    # In tuoi
    movl $4, %eax
    movl $1, %ebx
    movl $num_buffer, %ecx
    movl $2, %edx      # In 2 ky tu
    int $0x80

    # In thong bao du kien tot nghiep
    movl $4, %eax
    movl $1, %ebx
    movl $Du_kien_tot_nghiep, %ecx
    movl $Du_kien_tot_nghiep_length, %edx
    int $0x80

    # Tinh nam du kien tot nghiep
    movl NamSinh, %eax
    addl $22, %eax
    movl %eax, KetQua
    jmp print_ketqua

da_tot_nghiep:
    # In "Da tot nghiep"
    movl $4, %eax
    movl $1, %ebx
    movl $Da_tot_nghiep, %ecx
    movl $Da_tot_nghiep_length, %edx
    int $0x80
```

```
    # In "Tuoi: "
    movl $4, %eax
    movl $1, %ebx
    movl $Tuoi, %ecx
    movl $Tuoi_length, %edx
    int $0x80

    # Xu ly in tuoi
    movl TuoiHT, %eax
    movl $10, %ebx
    movl $0, %edx
    divl %ebx          # Lay chu so hang chuc
    addl $48, %eax     # Chuyen thanh ky tu
    movb %al, num_buffer
    addl $48, %edx     # Chuyen phan du thanh ky tu
    movb %dl, num_buffer + 1

    # In tuoi
    movl $4, %eax
    movl $1, %ebx
    movl $num_buffer, %ecx
    movl $2, %edx      # In 2 ky tu
    int $0x80

    # In thong bao nam da tot nghiep
    movl $4, %eax
    movl $1, %ebx
    movl $Nam_da_tot_nghiep, %ecx
    movl $Nam_da_tot_nghiep_length, %edx
    int $0x80

    # Lay nam da tot nghiep
    movl NamSinh, %eax
    addl $22, %eax
    movl %eax, KetQua

print_ketqua:
    #Chuyen so thanh chuoi
    movl $TempNum+3, %esi
    movl KetQua, %eax
.while:
    cmpl $0, %eax
    je .exit_while
    movl $0, %edx
    movl $10, %ebx
    divl %ebx
    addl $48, %edx
    addl %edx, (%esi)
    decl %esi
    jmp .while
.exit_while:
```

```
    # In so
    movl $4, %eax
    movl $1, %ebx
    movl $TempNum, %ecx
    movl $4, %edx
    int $0x80

    # In xuong dong
    movl $4, %eax
    movl $1, %ebx
    movl $newline, %ecx
    movl $1, %edx
    int $0x80

exit:
    movl $1, %eax
    movl $0, %ebx
    int $0x80
```

Giải thích:

Lưu đồ thuật toán:


- Giải thích một số hàm trong chương trình:
  - Hàm chuyển chuỗi thành số:

Giả sử chuỗi năm sinh được lưu trong mảng NamSinh có 4 phần tử, khi đó giá trị int của năm sinh sẽ được tính như sau:

```
int NamSinh=0;

For (int i=0; i<4; i++)

{

        NamSinh*=10;

        NamSinh+=NamSinh[i];

}
```

- Hàm chuyển số thành chuỗi:

Ở hàm này ta sẽ lấy đoạn hàm print_ketqua làm ví dụ, print_ketqua có chức năng in ra năm đã được tính trước đó. Tuy nhiên giá trị được lưu trong vùng nhớ ketqua là số nguyên nên ta phải chuyển dãy số đó thành dạng chuỗi để thực hiện in ra màn hình. Đoạn C minh họa hàm:

```
Char NamKetQua[4];

int i=0;

        While (ketqua!=0)

        {

                NamKetQua[i] = ketqua % 10;

                Ketqua /= 10;

                i++;

        }
```

Kết quả:

```
Nhap nam sinh gom 4 chu so: 1995
Da tot nghiep
Tuoi: 29
Nam da tot nghiep: 2017
```

```
Nhap nam sinh gom 4 chu so: 2008
Chua vao UIT
Tuoi: 16
Du kien vao UIT: 2026
```

```
Nhap nam sinh gom 4 chu so: 2006
Dang hoc tai UIT
Tuoi: 18
Du kien tot nghiep: 2028
```