ĐẠI HỌC BÁCH KHOA HÀ NỘI

TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

---📖---

BÁO CÁO GIỮA KỲ

THỰC HÀNH KIẾN TRÚC MÁY TÍNH

Giáo viên hướng dẫn: Hoàng Văn Hiệp

Sinh viên thực hiện: Nguyễn Thị Khánh Vân

MSSV: 20235869

Lớp: 156791 - IT3280

Bài A.4: Số hoàn hảo là số có giá trị bằng tổng các ước số không kể chính nó (ví dụ: 6 = 1 + 2 + 3, là số hoàn hảo). Viết hàm kiểm tra một số có phải là số hoàn hảo không. Nhập số nguyên dương N từ bàn phím, in ra màn hình các số hoàn hảo nhỏ hơn N.

***Ý tưởng:**

- Chương trình gồm 3 phần:
  - **Phần 1: Nhập số N từ bàn phím.** Đọc từng ký tự nhập vào (sử dụng lệnh nhập từng ký tự). Nhập từng ký tự vào và kiểm tra lần lượt, nếu '0' <= ký tự <= '9' thì chuyển các ký tự sang số rồi lấy: **số trước đó * 10 + số hiện tại = N**. Chương trình kết thúc khi nhập vào các ký tự khác ký tự chữ số. Vì yêu cầu đầu vào là một số nguyên dương nên số âm có ký tự '-' cũng kết thúc chương trình luôn.
  - **Phần 2: Thực hiện tính tổng các ước của các số nhỏ hơn N**. Kiểm tra từng lần lượt các số nguyên n từ 2 → N-1 để tính tổng các ước của n (không bao gồm n). Bắt đầu duyệt từ n = 2 vì 2 là số nhỏ nhất có ước thực sự (là 1), còn số 1 dù có ước là 1 nhưng số hoàn hảo lại không tính ước là chính nó
  - **Phần 3: Kiểm tra tổng các ước và in kết quả các số hoàn hảo**. Sau khi tính xong tổng ước của n thì ta kiểm tra xem có phải là số hoàn hảo không (s1 = n) rồi in luôn kết quả. Tăng dần n lên và tính tổng ước tương tự(ở phần 2).

* **Cách chương trình hoạt động**:

- Phần 1:
  - Sử dụng thanh ghi s0 để lưu giá trị số của N
  - Lưu mã ascii của '0' = 48 vào thanh ghi t2 để khi chuyển ký tự chữ số sang số thì chỉ việc trừ đi t2
  - Thực hiện vòng lặp để thực hiện nhập từng ký tự từ bàn phím
    - Sử dụng lệnh đọc 1 ký tự

```
li a7, 12              # Đọc 1 ký tự từ bàn phím
ecall
```

    - So sánh ký tự vừa nhập: '0' <= ký tự <= '9' .
      - Nếu ký tự nằm ngoài phạm vi này thì chương trình kết thúc.
      - Ngược lại, sử dụng lệnh trừ để chuyển ký tự sang số rồi cộng với kết quả trước đó đã được nhân 10 (s0*10)

```
sub a0, a0, t2          # Chuyển ký tự thành số
mul s0, s0, t3          # Nhân kết quả trước đó với 10
add s0, s0, a0          # Cộng giá trị số mới vào kết quả
```

  - Cứ thực hiện nhập từng ký tự đến khi gặp ký tự xuống dòng '\n' thì thực hiện dừng nhập ký tự

- Phần 2:
  - Thực hiện vòng lặp lần lượt từ n = 2 và tăng dần lên để kiểm tra tổng các ước
  - Để tìm ra tổng các ước s1 của n thì sử dụng vòng lặp từ i = 1→ $\sqrt{n}$ và kiểm tra n%i = 0 hay không
    - Nếu n%i ≠ 0 thì bỏ qua và tăng chỉ số i (i++)
    - Nếu n%i = 0 , n chia hết cho i, ta có thể tìm được ước còn lại của n bằng cách lấy n/i . Sau khi tìm được thì ta cộng 2 ước này vào tổng ước s1 của n. Ngoài ra, khi chia cho i =1 thì ước còn lại là n thì ta không cộng vào hoặc để tránh nghiệm bị trùng 2 lần thì ta cũng không cộng vào s1

```
beq t0, t3, next            # Nếu t0 = s0/t0 thì không cộng lại
beq t3, s0, next            # Nếu t3 = s0 (tức là ước là chính n)-> bỏ qua
```

    - Chương trình dừng lại khi duyệt hết
  - Tăng dần n lên để tính từng tổng ước. Vòng lặp dừng lại khi n = N
- Phần 3:
  - Sử dụng lệnh so sánh nếu bằng thì in kết quả. Ngược lại, ta bỏ qua và kiểm tra số tiếp theo n++
  - Kết thúc chương trình khi n = N

* **<u>Kết quả:</u>**

- **TH1**: Các trường hợp bình thường
  - N = 6

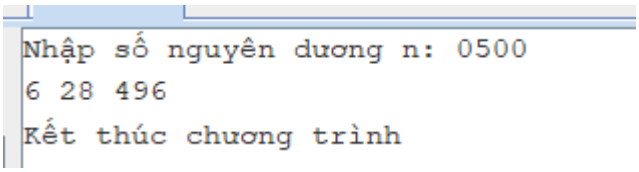
```
Nhập số nguyên dương n: 6

Kết thúc chương trình
```

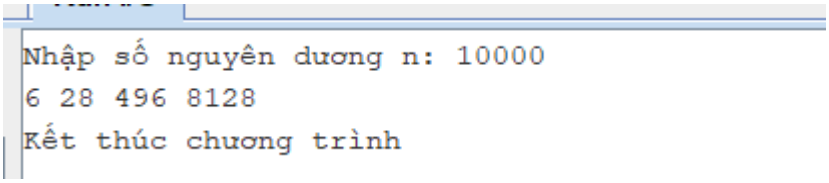
  - N = 100

```
Nhập số nguyên dương n: 100
6 28
Kết thúc chương trình
```

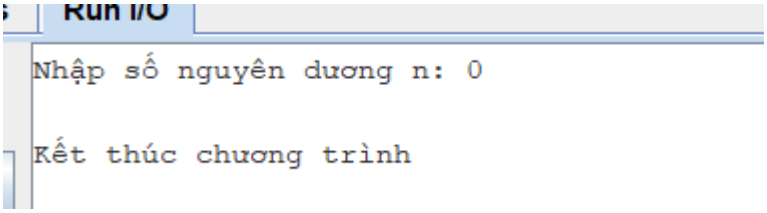
- N = 500



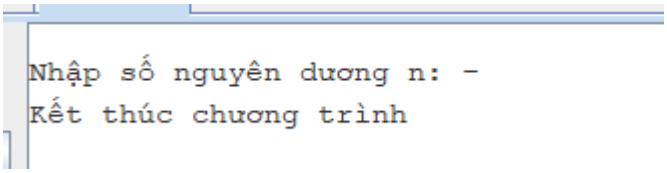
- N = 10000

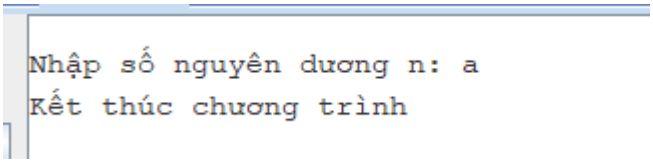


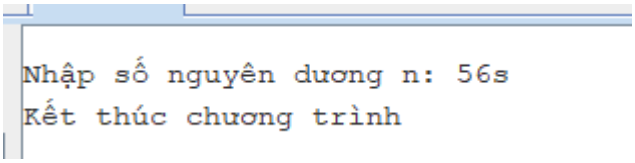
- **TH2**: Nhập vào số không phải số nguyên dương
  - N = 0:



  - N là chữ cái hoặc số âm







Bài B.10: Nhập mảng số nguyên từ bàn phím. In ra màn hình vị trí và giá trị của phần tử âm lớn nhất trong mảng.

*** Ý tưởng:**

- Chương trình gồm 3 phần:
  - **Phần 1: Nhập các số đầu vào**. Giới hạn số phần tử mảng, sử dụng mảng để lưu các phần tử nhập vào từ bàn phím
  - **Phần 2: Tìm ra số âm lớn nhất trong mảng**. Sử dụng 1 thanh ghi lưu giá trị nhỏ nhất trong risc-v và lấy từng phần tử trong mảng ra so sánh với nó. Nếu số đó âm và lớn hơn hoặc bằng giá trị âm lớn nhất thì cập nhật.
  - **Phần 3: Tìm vị trí và in kết quả số âm lớn nhất.** Nếu như mảng có nhiều hơn 1 vị trí của phần tử âm lớn nhất thì duyệt lại mảng 1 lần nữa để in ra các vị trí của nó.

*** Cách chương trình hoạt động:**

Phần 1:

- Đầu tiên là nhập số phần tử trong mảng theo mong muốn với điều kiện mảng có tối đa 20 phần tử: 0< N < 21
  - Nếu nhập số phần tử ngoài khoảng cho phép thì chương trình in ra 'đầu vào không hợp lệ' và kết thúc
  - Nhập vào N là số nguyên đúng theo điều kiện thì tiếp tục nhập vào các phần tử trong mảng
- Tiếp theo là nhập các số nguyên cho mảng. Sử dụng vòng lặp để nhập vào và đẩy từng số nguyên vào trong mảng A (kiểu word) đã được khai báo. Mỗi lần nhập vào 1 số thì phải cộng thêm 4 byte vào địa chỉ mảng vì kiểu word là 4 byte

```
li a7, 5                # Nhập các phần tử số vào mảng A
ecall
sw a0, 0(a1)            # Thêm số vừa nhập vào mảng A

addi t0, t0, 1          # Tăng chỉ số thêm 1
addi a1, a1, 4          # 1 phần tử là 4 byte
```

- Chương trình dừng nhập khi đã nhập đủ số lượng các phần tử

Phần 2:

- Sử dụng thanh ghi s0 để chứa số âm lớn nhất trong mảng. Khai báo s0 với giá trị nhỏ nhất trong risc-v là -2^{31} (s0 = -2147483648)
- Lấy thanh ghi t1 để đánh dấu xem trong mảng có tồn tại số âm lớn nhất không. Đặt t1 = -1 (trước khi kiểm tra)

- Sử dụng vòng lặp từ i = 0 → N - 1 để duyệt qua các phần tử trong mảng
    - Lấy từng phần tử trong mảng ra so sánh
        - Nếu như có số âm lớn hơn hoặc bằng s0 thì đặt lại t1 = 1 nghĩa là có tồn tại số âm lớn nhất đồng thời cập nhật s0

```
addi s0, s1, 0          # Cập nhật phần tử âm
li t1, 1                # Cập nhật có tồn tại phần tử âm lớn nhất
```

        - Ngược lại, bỏ qua và tiếp tục tăng chỉ số i++
    - Chương trình dừng lại khi đã duyệt hết mảng

Phần 3:

- Kiểm tra t1 < 0 → Không tồn tại số âm lớn nhất, kết thúc chương trình. Nếu t1 > 0 → In ra số âm lớn nhất

```
    bltz t1, end            # Nếu t1 âm -> Không tồn tại phần tử âm lớn nhất
```

- Sử dụng vòng lặp để duyệt lại mảng 1 lần nữa để in ra các vị trí của số âm lớn nhất nếu như có nhiều hơn 1 vị trí
    - Đặt lại địa chỉ đầu mảng và chỉ số i = 0
    - Lấy từng phần tử ra và so sánh với s0
        - Nếu như bằng thì in ra vị trí
        - Ngược lại thì bỏ qua và tiếp tục kiểm tra phần tử kế tiếp

```
        lw s1, 0(a1)            # Lấy từng phần tử trong mảng A
        bne s1, s0, ship        # Nếu như s1 khác s0 -> duyệt phần tử kế tiếp

print:
        li a7, 1                # In chỉ số của phần tử âm lớn nhất
        addi a0, t0, 0
        ecall
```

- Khi duyệt hết các phần tử thì kết thúc chương trình.

<u>* Kết quả:</u>

- **TH1:** Nhập vào số lượng phần tử N <= 0 hoặc N >= 21

```
Nhập số phần tử của mảng(0<N<21): 0
ERROR!
Đầu vào không hợp lệ
```

```
Nhập số phần tử của mảng(0<N<21): -2
ERROR!
Đầu vào không hợp lệ
```

```
Nhập số phần tử của mảng(0<N<21): 21
ERROR!
Đầu vào không hợp lệ
```

- **TH2:** N > 0
  - *TH2.1*: mảng toàn số dương

```
Nhập số phần tử của mảng(0<N<21): 5
Nhập số: 1
2
3
4
5
Không tồn tại phần tử âm lớn nhất !
```

  - *TH2.2:* mảng toàn số âm

```
Nhập số phần tử của mảng(0<N<21): 5
Nhập số: -1
-2
-3
-4
-5

Giá trị phần tử âm lớn nhất: -1
Vị trí của phần tử âm lớn nhất: 0;
```

  - *TH2.3:* mảng có cả số âm và dương

```
Nhập số phần tử của mảng(0<N<21): 5
Nhập số: 2
-4
2
3
-1

Giá trị phần tử âm lớn nhất: -1
Vị trí của phần tử âm lớn nhất: 4;
```

- *TH2.4:* phần tử âm lớn nhất trong mảng có nhiều vị trí

```
Nhập số phần tử của mảng(0<N<21): 5
Nhập số: -3
0
7
-3
-8

Giá trị phần tử âm lớn nhất: -3
Vị trí của phần tử âm lớn nhất: 0;3;
```

Bài C.15: Nhập vào 2 xâu ký tự A và B. In ra màn hình các ký tự chữ số không xuất hiện cả trong A và B.

***Ý tưởng:**

- Chương trình bao gồm 2 phần:
  - **Phần 1: Thực hiện nhập vào 2 xâu ký tự. Đồng thời đẩy các ký tự số vào trong mảng Digits**. Theo yêu cầu của đề bài là in ra các ký tự số không xuất hiện trong cả 2 chuỗi. Nên thay vì tạo 2 mảng để lưu 2 chuỗi nhập vào thì ta có thể tạo 1 mảng Digits để lưu các ký tự số xuất hiện trong cả hai chuỗi. Để nhập vào 2 chuỗi thì ta chỉ việc thực hiện hai vòng nhập liên tiếp. Các ký tự nhập vào chỉ lấy các ký tự số để lưu vào mảng Digits mà thôi. Việc này đảm bảo cho mảng Digits chứa toàn bộ các ký tự của cả 2 chuỗi A và B.
  - **Phần 2: Kiểm tra xem trong mảng Digits không có số nào thì in ra.** Sau khi đã lưu toàn bộ các ký tự số vào trong mảng Digits, chương trình thực hiện kiểm tra các số từ '0' đến '9' có xuất hiện trong mảng Digits hay không. Nếu không xuất hiện thì in số đó ra màn hình.

* **Cách chương trình hoạt động**:

- Phần 1:
  - Giới hạn 20 ký tự cho 2 chuỗi nhập vào, lấy địa chỉ của đầu mảng Digits(.space) cho vào thanh ghi a1, đồng thời sử dụng thanh ghi t4 để đếm số ký tự trong mảng Digits
  - Sau khi in "Nhập chuỗi A (tối đa 20 ký tự): " dùng lệnh jal để nhảy đến chương trình con thực hiện việc việc nhập từng ký tự
    - Sử dụng lệnh nhập từng ký tự vào

- Kiểm tra ký tự đó có nằm trong khoảng ‘0’ đến ‘9’ hay không. Nếu như nằm trong thì thêm vào mảng Digits và tăng chỉ số mảng Digits

```
li t1, 48                       # Giá trị ASCII của '0'
blt a0, t1, skip                # Nếu nhỏ hơn '0' -> bỏ qua

li t1, 57                       # Giá trị ASCII của '9'
bgt a0, t1, skip                # Nếu lớn hơn '9' -> bỏ qua

# Lưu toàn bộ chữ số từ cả chuỗi A và B vào chung mảng Digits
add t1, t4, a1                  # t1 = i + Digits[0]
sb a0, 0(t1)                    # a0 = Digits[i]

addi t4, t4, 1
```

- Ngược lại, ta bỏ qua và tiếp tục nhập ký tự tiếp theo
- Chương trình con dừng việc nhập ký tự khi đạt đủ 20 ký tự hoặc gặp dấu xuống dòng ‘\n’ → sử dụng lệnh jr ra để quay về chương trình chính ở địa chỉ tiếp theo ngay sau lệnh jal loop

- Khi quay lại chương trình chính, in xâu “\nNhập chuỗi B (tối đa 20 ký tự): ” và tiếp tục thực hiện nhập ký tự của chuỗi B tương tự như chuỗi A
- Sau 2 lần nhảy đến chương trình con, mảng Digits lưu những ký tự số có trong cả 2 chuỗi A và B

- Phần 2:
  - Kiểm tra t4 (thanh ghi đếm số ký tự trong Digits).
    - Nếu t4 = 0 → mảng Digits trống → nhảy đến nhãn in ra tất cả các số từ ‘0’ đến ‘9’
    - T4 > 0 → sử dụng vòng lặp để kiểm tra từng số hết mảng Digits bắt đầu lần lượt từ ‘0’ đến ‘9’
      - Lấy từng ký tự trong mảng Digits ra rồi so sánh với ‘0’. Nếu số xuất hiện thì kiểm tra số tiếp theo ‘1’ cứ tiếp tục cho đến hết số ‘9’

```
add t3, t2, a1                  # Lấy địa chỉ của các ký tự số trong Digits
lb t5, 0(t3)

addi t1, t0, 48                 # Các số từ 0->9
beq t5, t1, skip_loop           # Nếu xuất hiện thì kiểm tra số tiếp theo
```

      - Nếu như duyệt hết mảng Digits mà số chưa xuất hiện thì thực hiện in ra số đó
      - Tiếp tục kiểm tra các số tiếp theo

➢ Chương trình kết thúc khi duyệt hết tất cả 10 số

*** Kết quả:**

- *TH1*: Không nhập gì

```
Nhập chuỗi A (tối đa 20 ký tự):

Nhập chuỗi B (tối đa 20 ký tự):

Các ký tự chữ số không xuất hiện cả trong A và B: 0;1;2;3;4;5;6;7;8;9;
```

- *TH2*: Chỉ nhập vào 2 mảng các ký tự chữ cái

```
Nhập chuỗi A (tối đa 20 ký tự): afs

Nhập chuỗi B (tối đa 20 ký tự): asawe

Các ký tự chữ số không xuất hiện cả trong A và B: 0;1;2;3;4;5;6;7;8;9;
```

- *TH3*: Chỉ nhập vào 2 mảng các ký tự chữ số

```
Nhập chuỗi A (tối đa 20 ký tự): 651

Nhập chuỗi B (tối đa 20 ký tự): 543

Các ký tự chữ số không xuất hiện cả trong A và B: 0;2;7;8;9;
```

- *TH4*: Cả 2 chuỗi có cả chữ và số

```
Nhập chuỗi A (tối đa 20 ký tự): afsgwe013

Nhập chuỗi B (tối đa 20 ký tự): asgj4251

Các ký tự chữ số không xuất hiện cả trong A và B: 6;7;8;9;
```

- *TH5*: 2 mảng có đủ 10 số

```
Nhập chuỗi A (tối đa 20 ký tự): 123870

Nhập chuỗi B (tối đa 20 ký tự): 5469124afax

Các ký tự chữ số không xuất hiện cả trong A và B
```