**Bài tập lập trình assembly.**

***Note:*** Với những bài phức tạp, nên viết giải thuật sử dụng [flowchat](https://www.draw.io/) hoặc pseodocode.

Làm xong bài nào thì copy kết quả vào file này để lưu trữ.

For all exams, you should make use of the [Beta documentation](https://prod-edxapp.edx-cdn.org/assets/courseware/v1/a31badb240163a3d3e3f1573af782c79/asset-v1:MITx+6.004.2x_2+3T2016+type@asset+block/pdfs_course_beta.pdf) and you can use the [Bsim](http://computationstructures.org/exercises/sandboxes/bsim.html) (Beta simulator) to verify the exams.

------------------------------------------

**1.** Interchange two parts from a[1] and a[3].

**2.** Write a program to do the calculation: a[0]=a[1]+a[2]

**3.** Divide a[0] by a[1], the integer part stored in a[2], the remaining part stored in a[3].

**4.** a[0] store the value of minutes. Convert the number of minutes to hour and store in a[1] and store the remaining one in a[2]. (For example: a[0] = 125, then a[1] = 2 and a[2] = 5).

**5.** a [0] store the value of seconds. Convert the number of minutes to hour and store in a[1] and store the remaining minutes in a[2] and the remaining seconds in a[3]. (For example: a[0] = 3745 then a[1] = 1 and a[2] = 2.

**6.** If a[1] > a[3], then interchange.

**7.** Find the minimum value of a[0], a[1] and a[2], the result is stored in a[3].

**8.** Find the maximum value of a[0], a[1] and a[2], the result is stored in a[3].

**9.** Check if a[0], a[1] and a[2] are the lengths of the triangle. If it is true, store 1 to a[3], otherwise store 0.

**10.** Sort a[0], a[1] and a[2] in ascending order.

**11.** Sort a[0], a[1] and a[2] in descending order.

**13.** Calculate the sum s = 1 + 2 + ... + a[0] and store it to register r0.

**14.** Calculate the total number of array elements from a[0] to a[9] and store it to a[10].

**15.** Find the largest number of array elements from a[0] to a[9] and store it to a[10].

**16.** Find the largest common divisor of a[0] and a[1] and store it to a[2].

**17.** Find the smallest common multiple of a[0] and a[1] and store it to a[2].

**18.** Determine if a[0] is prime number or not. If yes, write 1 to a[1], otherwise write 0.

**19.** Find the smallest prime number that is greater than a[0], the result is stored in a[1]. (For example, a[0]=8 then a[1]=11).

**20.** Determine if a[0] is a square number or not. If yes, write 1 to a[1], otherwise write 0.

Project - Design Problem: Bubble sort

Write 5 programs using the Assemby language to perform the following tasks:

**Program 1:** Write a program to swap two array elements A[5] and A[6] in Assembly language.

**Program 2:** Write a program to swap two array elements A[i] and A[i+1] in Assembly language.

**Program 3:** Write a program to swap two array elements A[i] and A[i+1] if A[i]>A[i+1] in Assembly language. (increasing order)

**Program 4:** Write a program to sort a data array in increasing order using the "bubble\_sort" algorithm.

**Program5:** Write a program to sort a data array in program 4 using the “selection sort” algorithm.

Template 1:

/\* Khai bao mảng

Load (LD): Giá trị trong ô nhớ tải vào thanh ghi

Store (ST): Giá trị trong thanh ghi lưu vào ô nhớ

Đề bài: Ghi giá trị 0x123 vào a[3]

Tải giá trị a[3] vao thanh ghi r10 \*/

.include "beta.uasm"

// Viết code vào đây

HALT()

a: LONG(9) // a[0] – Address a

LONG(2) // a[1] – Address a+4 = a+4\*1

LONG(0) // a[2] – Address a+8 = a+4\*2

LONG(4) // a[3] – Address a+12 = a+4\*3

LONG(5) // …

Template 2:

// Đề bài: Đổi chỗ 2 phần tử mảng a[3] và a[5]

.include "beta.uasm"

// Viết code vào đây

HALT()

a: LONG(9) // a[0] – Address a

LONG(2) // a[1] – Address a+4 = a+4\*1

LONG(0) // a[2] – Address a+8 = a+4\*2

LONG(4) // a[3] – Address a+12 = a+4\*3

LONG(5) // …

Template 3:

// Đề bài: Nếu a[1] > a[3] thì đổi chỗ

.include "beta.uasm"

HALT()

a: LONG(9) // a[0] – Address a

LONG(2) // a[1] – Address a+4 = a+4\*1

LONG(0) // a[2] – Address a+8 = a+4\*2

LONG(4) // a[3] – Address a+12 = a+4\*3

LONG(5) // a[4] - …

Program :

/\*

Ví dụ 1:

r0=5

r1=7

r2=r0+r1

\*/

.include "beta.uasm"

ADDC(r31,5,r0) // r0=r31+5

ADDC(r31,7,r1)

ADD(r0,r1,r2) // r0+r1-->r2;

HALT()

Program :

/\*

Ví dụ 2:

r0=25

r1=7

r2=r0/r1

\*/

.include "beta.uasm"

ADDC(r31,25,r0)

ADDC(r31,7,r1)

DIV(r0,r1,r2) // r2=r0/r1

HALT()

Program :

Program :

Program :

Program :

Program :

Program :

Program :

Program :

Program :

Program :

Program :