HSTI
Hardware e Software das Tecnologias de Informação
(Hardware and Software of Information Technologies)

Licenciatura em Sistemas e Tecnologias de Informação
Nova IMS
Assembly Programming Project for 2021/2022

Preliminary notes

Each **group of 2 students** must choose one of the proposed projects, and inform the teacher responsible (Eng. Pedro Castro Fernandes – **pmfernandes@novaims.unl.pt**) so as to avoid having repetitions. It is recommended that you **suggest 3 or 4 projects in order**, in case the preferred one is already taken. The projects have very different difficulty levels, and the easier ones have a maximum grade that is less than 20/20.

For ALL projects you must write:

1) A **sub-routine** to implement the desired function;

2) A **"main" program** that calls that routine, if necessary passing parameters to that routine.

Provisions must be made so that, during the discussion, the teacher can change the values of the parameters passed to the routine.

All projects require that a final **report** be presented, and this report must include:

A) Objective – A description of the objective of the project

B) Fluxograms (and block diagram if necessary) – A detailed fluxogram (or flowchart) of each routine developed.

C) Memory Map and the use given to each of the registers of the microprocessor, identifying clearly where each variable is stored.

D) Listing of all code with comments

E) Conclusions

F) An annex with the machine code and addresses used.

All groups must present their projects, showing the program running on a simulator or a development Kit (or any other computer with a 8085), and send, together with the report, an electronic version of the program in HEX format, or as a text file that can be easily assembled by cutting and pasting it into an assembler program. The reports and code should be sent in electronic version, in a **zip file** that should have the student number (of the student with lowest number) followed by underscore ("_") followed by whatever you want (ex: *20201234_OrderInt8Vector*), up to the **4th of January**.

Projects

1. **Calculate the sum of two numbers.** Write a routine that computes the sum of two bytes. The two bytes are passed to the routine in registers A and C, and returned in register A. **(Max=11)**

2. **Compute the complement of a 8 bit number.** Write a routine that computes the complement of a number (i.e., changes all 0's by 1's, and vice-versa). The number is passed in register A, and the result should be left also in register A. **(Max=11)**

3. **Exchange the first nibble (4 bits) with the second nibble.** Write a routine that exchanges the two nibbles of a byte. The byte is passed in register A, and the result should be left also in register A. **(Max=11)**

4. **Multiply a number by two.** Write a routine that multiplies a number by two. The number to be divided is passed in register A, and the result should be left also in register A. **(Max=11)**

5. **Divide a number by two.** Write a routine that divides a number by two. The number to be divided is passed in register A, and the result should be left also in register A. **(Max=11)**

6. **Subtract two numbers.** Write a routine that subtracts one number from another. The two bytes are passed to the routine in registers A and B, and B-A must be returned in register A. **(Max=12)**

7. **Force the least significant bit of a byte to 0.** Write a routine that forces the least significan bit of a byte to 0. The byte is passed in register A, and the result should be left also in register A. **(Max=12)**

8. **Force the least significant bit of a byte to 1.** Write a routine that forces the least significan bit of a byte to 0. The byte is passed in register A, and the result should be left also in register A. **(Max=12)**

9. Write the assembler code that corresponds to the following C code **(Max=13)**:
10.
11.

```
xpto=1;
var1=0;
var2=xpto;
if( var1==var2)
  var1=2;
else
  var1=3;
xpto=4;
```

```
A=0;
B=0;
for(i=1;i>0;i--);
 {
   A=A+i:
   B=B+2*i:
 }
```

```
stop=127;
start=1;
num=0
while(start<stop)
 {
   start=start+start+10;
   num=num+1;
 }
```

12. **Calculate the difference bewteen two numbers.** Write a routine that computes the difference between two bytes. The two bytes are passed to the routine in registers D and E, and the difference (always a positive number) is returned in register A. **(Max=14)**

13. **Calculate the average of four numbers.** Write a routine that computes the average of the four numbers stored in registers B,C,D and E, and leaves the result in register A. **(Max=14)**

14. **Multiplication of a number by 10,** without verifications**.** Write a routine that receives in the accumulator a number that will be multiplied by 10. The result should be left in the accumulator, and you can assume that no errors will occur (i.e., the result will not exceed 255). **(Max=14)**

15. **Division of a number by 10.** Write a routine that receives in the accumulator a number that will be divided by 10. The result should be left in the accumulator, and consider just the integer part of the division. **(Max=14)**

16. **Copy a block of data from one location in memory to another one.** Write a routine that copies a set of bytes from one location in memory to another one. The address of the first byte to be copied is passed in register pair DE, and the first destination address is passed in register pair HL. The number of bytes to be copied in passed in register C. **(Max=14)**

17. **Find two consecutive and equal bytes in a vector.** Write a routine that receives in register pair HL the 1st address of a vector of bytes, in register A the length of the vector, and returns in the accumulator the value "1" if there are two consecutive bytes with the same value, and "0" otherwise. **(Max=14)**

18. **Multiply two 8 bit numbers, leaving the result in a 16bit register pair.** Write a routine that receives in registers D and E two 8 bit numbers, and returns in register pair HL their product. **(Max=14)**

19. **Division of two 8 bit numbers.** Write a routine that receives in registers B and C two 8 bit numbers, and returns in the accumulator B/D (only the integer part). **(Max=14)**

20. **Compute the negative of a black and white image.** Assume that you have in a given memory address a bitmap of a black and white image, with 1 bit color depth, where a pixel value of 1 indicates that it is black, and 0 indicates that it is white. Write a routine that receives in register pair HL the address of the first byte of the image and in register C the number of bits, and then exchanges all 0 by 1 and vice-versa. **(Max=14)**

21. **Multiplication of 8 bit numbers.** Write a routine that receives in registers D and E two 8 bit numbers, and returns in the accumulator their product. If the result cannot be stored in 8 bits, the routine must leave the value 0 (zero) in the accumulator. **(Max=14)**

22. **Change to uppercase.** Write a routine that receives in register pair HL the first address of a text with length *N* (in characters), where N is stored in register A, and changes all lowercase characters into uppercase (assuming that the text is written using ASCII code). **(Max=14)**

23. **Change to lowercase.** Write a routine that receives in register pair HL the first address of a text with length *N* (in characters), where N is stored in register A, and changes all uppercase characters into lowercase (assuming that the text is written using ASCII code). **(Max=14)**

24. **Multiplication of 16 bit numbers.** Write a routine that receives in register pairs BC and HL two 16 bit numbers, and returns in register pair HL their product. If the result cannot be stored in 16 bits, the routine must leave the value 0 (zero) in the accumulator (and 1 otherwise).**(Max=14)**

25. **Multiplication of 16 bit numbers, leaving the result in 32 bits.** Write a routine that receives in register pairs BC and HL two 16 bit numbers, and returns in register pair HL the lower (least significant) part and in in DE the higher (most significant) part of their product. **(Max=15)**

26. **Multiplication of a number by 10.** Write a routine that receives in the accumulator a number that will be multiplied by 10. The result should be left in the accumulator, but if it exceeds 255, then the accumulator should be returned with the value 0. **(Max=15)**

27. **Find the maximum in a vector of bytes.** Write a routine that receives in register pair HL the first address of a vector of bytes, and in the accumulator it's length. The routine must then find the maximum, and store it's value in register C and it's position in register DE. **(Max=16)**

28. **Find the minimum in a vector of bytes.** Write a routine that receives in register pair HL the first address of a vector of bytes, and in the accumulator it's length. The routine must then find the minimum, and store it's value in register C and it's position in register DE. **(Max=16)**

29. **Order the values of the registers.** Write a routine that orders all the registers of the microprocessor (A,B,C,D,E,H, and L), so that A has the highest value, and L the lowest. **(Max=16)**

30. **Sum two astronomical numbers of the same length.** In some cases it is necessary to sum very large numbers, that may be represented by any number of bytes (these numbers are called "astronomical numbers in computing). Write a routine that receives in register pairs HL and DE the first address of these two numbers, and in the accumulator the number of bytes that these numbers have. The routine should leave the result in the addresses originally pointed to by HL. Assume that the numbers are stored in bigendian format.**(Max=18)**

31. **Multiplication of 32 bit numbers, leaving the result in 64 bits.** Write a routine to multiply who 32 bit numbers that receives in register pairs DE and HL the addresses of two 32 bit numbers, and in register pair BC the address where the result should be left. **(Max=18)**

32. **Compute powers.** Write a routine that receives in registers B and C two 8 bit numbers, and returns in the accumulator B^C (B to the power C). If the result cannot be stored in the accumulator, the routine should leave 0 (zero) in the accumulator.**(Max=18)**

33. **Division of 16 bit numbers.** Write a routine that receives in register pairs DE and HL two 16 bit numbers, and returns in register pair BC DE/HL (only the integer part). **(Max=18)**

34. **Division of 8 bit numbers, including the non-integer part..** Write a routine that receives in registers B and C two 8 bit numbers, and returns in register pair DE B/C, with the integer part in D and the non-integer part in E. **(Max=18)**

35. **Order vectors of bytes.** Write a routine that receives in register pair HL the first address of a vector of bytes, and in the accumulator it's length. The routine must then order the vector, in ascending order (i.e., lowest number first). **(Max=18)**

36. **Order vectors of 16 bit numbers.** Write a routine that receives in register pair HL the first address of a vector of 16b bit numbers, and in the accumulator it's length. The routine must then order the vector, in ascending order (i.e., lowest number first). **(Max=20)**

37. **Factoring numbers.** Write a routine to factor numbers, that receives in the accumulator a number, and in register pair HL the address of the positions in memory where the factors must be stored. **(Max=20)**

38. **Break a Cesar code.** Imagine that a dangerous terrorist is communicating through the internet, but is cyphering his messages with a Cesar code. In a Cesar code, a given number (called key) is added to the codes of all the characters that are sent. We know that the terrorist is targeting the Portuguese Air Force Base at OTA, so the word OTA will certainly occur in the message. We want to write a program to detect that word (OTA) even when an unknown number is added to all it's letters. When the word is detected, the program should indicate what was the key used to cypher it, and decode the whole message with it. In the end, the program should leave the decoded message in the same place where the original message was, and leave in the accumulator the key used. **(Max=20)**

39. **Access to a database.** Imagine that you have a database where each record has 12 bytes to store the name of a person, and 4 bytes to store their telephone number. You need to write a routine in assembly that finds the telephone number of a person, given the name of that person. The routine should receive in HL the address of the first record in the database, in DE the memory address where the name to search is stored, and in the accumulator the number of records in the database. The routine should return in DEHL their telephone number (in register pair DE the most significant part and in HL the least significant), and the value 1 in the accumulator. If the name of the person is not found in the database, the routine should return FF in the accumulator. **(Max=20)**

The following problems use assembler code, but on a x86 computer, and interacting with a high-level language, using "Inline Assembler". This feature can be found, for example, on Microsoft's Visual Studio, using C++. To do these projects you must familiarize yourself with x86 machine code instructions (and the x86 registers available on your processor) and pass parameters from the high level language of your choice to assembler. All these programs have a maximum grade 20. **(Max=20)**

40. **Integer division of a number by a power of 2**. Write a routine that receives two parameters (CX and BX) and returns $AX*2^{BX}$.

41. **Factorial of a number**. Write a routine that computes the factorial of a number.

42. **Write any of the routines specified in points 9 to 39**. Instead of BC use BX, instead of DE use DX, and instead of HL use CX. In these cases, the maximum grade is the one of the original problem, plus 2.