

- **Número USP:** 11726111
- **Nome completo do aluno:** Santiago Quintero Hincapié

Primeira questão

A funcionalidade deste program é: mostrar na saída o código ASCII de uma sequência de letras, neste caso, os dados conseguidos com meu nome do item 2 da preparação dos dados, ou seja:

Hexa	letra	Decimal
<u>53</u>	S	83
<u>41</u>	A	65
<u>4E</u>	N	78
<u>54</u>	T	84
<u>49</u>	I	73
<u>41</u>	A	65
<u>47</u>	G	71
<u>4F</u>	O	79
<u>53</u>	S	83
<u>41</u>	A	65
<u>4E</u>	N	78
<u>54</u>	T	84
<u>49</u>	I	73
<u>41</u>	A	65
<u>47</u>	G	71
<u>4F</u>	O	79

O código objeto carregável é

Tabela do Programa:

IC	CO	OP	Rótulo	
100	LD	f00	STAR	Iniciar o contador
102	MM	201		
104	LD	202	IN	Iniciar a instrução LD do endereço 10c
106	MM	10c		
108	LD	203		
10a	MM	10d		
10c	LD	f01	LDA	Mostrar o dado correspondente á letra na saída
10e	PD	004		
110	LD	201		Atualizar o contador e perguntar se é zero
112	-	200		
114	MM	201		
116	JZ	12a		
118	LD	10d		Atualizar a instrução LD do endereço 10c
11a	+	200		
11c	MM	10d		
11e	JZ	122		Jump do loop
120	JP	10c		
122	LD	10c	INCR	Atualizar a instrução LD do endereço 10c
124	+	200		
126	MM	10c		
128	JP	10c		
12a	HM	100	FORA	Terminar

Tabela de dados: esta tabela trabalha com números decimais, por isso transformei os numero hexadecimais em números decimais.

200	1	//UM	
201	0	//CONTADOR	
202	143	//8f em hexa	Instrução LDA INIC
203	1	//01 em hexa	
f00	16	//comprimento do texto	
f01	83	//53 em hexa "S"	
f02	65	//41 em hexa "A"	
f03	78	//4E em hexa "N"	
f04	84	//54 em hexa "T"	
f05	73	//49 em hexa "I"	
f06	65	//41 em hexa "A"	
f07	71	//47 em hexa "G"	
f08	79	//4F em hexa "O"	
f09	83	//53 em hexa "S"	
f0a	65	//41 em hexa "A"	
f0b	78	//4E em hexa "N"	
f0c	84	//54 em hexa "T"	
f0d	73	//49 em hexa "I"	
f0e	65	//41 em hexa "A"	
f10	71	//47 em hexa "G"	
f11	79	//4F em hexa "O"	

Implementação do código objeto na MVN

Primeramente escrevemos os dados e as instruções no programa:

The interface displays the following components:

- Instruction Table:** A table with columns CI, CO, and OP. It lists instructions from address 100 to 122.
- Data Table:** A table with columns Endereco and Dado. It lists data from address f01 to f06.
- Memory Dump:** A grid showing memory addresses from 00 to 21 and their corresponding hexadecimal values (mostly XX).
- Control Panel:** Includes fields for Entrada, Endereço (set to 100), Saida, and Acomulador (set to 00), along with buttons for LOAD, Next Step, and Reset.

Red annotations highlight the following elements:

- A red box around the instruction table.
- A red box around the data table.
- A red box around the "Endereço: 100" field, with an arrow pointing to it and the text "endereço inicial".
- A red arrow pointing to the memory dump at address 07, with the text "tabela do programa".
- A red arrow pointing to the memory dump at address 14, with the text "tabela de dados".

Logo damos click em *LOAD* para montar tudo na memoria:

CI	CO	OP
040	X	XXX
042	X	XXX
044	X	XXX
046	X	XXX
048	X	XXX
04a	X	XXX
04c	X	XXX
04e	X	XXX
050	X	XXX
052	X	XXX
054	X	XXX
056	X	XXX
058	X	XXX
05a	X	XXX
05c	X	XXX
05e	X	XXX
060	X	XXX
062	X	XXX

Endereco	Dado
XXX	XX
XXX	XX
XXX	XX
XXX	XX
XXX	XX
XXX	XX

Entrada:

Endereço:

Saída:

Acomulador:

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
00	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
01	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
02	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
03	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
04	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
05	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
06	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
07	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
08	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
09	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
0a	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
0b	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
0c	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
0d	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
0e	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
0f	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
10	8f	00	92	01	82	02	91	0c	82	03	91	0d	8f	01	e0	04
11	82	01	52	00	92	01	11	2a	81	0d	42	00	91	0d	11	22
12	01	0c	81	0c	42	00	91	0c	01	0c	c1	00	XX	XX	XX	XX
13	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
14	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
15	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
16	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
17	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
18	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
19	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
1a	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
1b	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
1c	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
1d	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
1e	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
1f	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
20	01	00	8f	01	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
21	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Pode-se ver parte do código de maquina na memoria.

Executando a primeira linha (click em Next Step): LD f00

The screenshot shows a debugger window with several panels. On the left, there is a table with columns 'CI', 'CO', and 'OP'. Below it is another table with columns 'Endereco' and 'Dado'. At the bottom left, there are input fields for 'Entrada:', 'Endereço:', 'Saída:', and 'Acomulador:', along with buttons for 'LOAD', 'RUN', 'Next Step', and 'Reset'. The 'Acomulador' field is highlighted with a red box and contains the value '10'. On the right side of the window, there is a large memory dump table with columns for hexadecimal addresses (00 to 21) and hexadecimal data (00 to ff). The data in the memory dump is mostly 'XX', except for a few rows (10, 11, 20, 21) which contain specific hexadecimal values.

CI	CO	OP
040	X	XXX
042	X	XXX
044	X	XXX
046	X	XXX
048	X	XXX
04a	X	XXX
04c	X	XXX
04e	X	XXX
050	X	XXX
052	X	XXX
054	X	XXX
056	X	XXX
058	X	XXX
05a	X	XXX
05c	X	XXX
05e	X	XXX
060	X	XXX
062	X	XXX

Endereco	Dado
XXX	XX
XXX	XX
XXX	XX
XXX	XX
XXX	XX
XXX	XX

Entrada: LOAD

Endereço: RUN

Saída: Next Step

Acomulador: Reset

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
00	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
01	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
02	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
03	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
04	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
05	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
06	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
07	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
08	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
09	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
0a	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
0b	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
0c	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
0d	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
0e	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
0f	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
10	8f	00	92	01	82	02	91	0c	82	03	91	0d	8f	01	e0	04
11	82	01	52	00	92	01	11	2a	81	0d	42	00	91	0d	11	22
12	01	0c	81	0c	42	00	91	0c	01	0c	c1	00	XX	XX	XX	XX
13	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
14	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
15	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
16	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
17	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
18	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
19	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
1a	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
1b	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
1c	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
1d	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
1e	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
1f	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
20	01	00	8f	01	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
21	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

O acumulador pega o que esta no endereço f00, ou seja, o comprimento do vector de letras o qual é 16 (o acumulador sempre mostra tudo em código de maquina, ou seja, em hexadecimal).

Executando a próxima linha (click em Next Step): MM 201

The screenshot shows a debugger window with the following components:

- Register List:** A table with columns CI, CO, and OP. It lists registers from 040 to 062. Most have 'X' in CO and 'XXX' in OP.
- Register Data Table:** A table with columns Endereco and Dado. It shows five rows, each with 'XXX' in Endereco and 'XX' in Dado.
- Input/Output Section:**
 - Entrada: An empty text box with a 'LOAD' button.
 - Endereço: A text box containing '100' with a 'RUN' button.
 - Saída: An empty text box with a 'Next Step' button (highlighted with a red border).
 - Acomulador: A text box containing '10' with a 'Reset' button.
- Memory Dump:** A large grid showing memory addresses from 00 to 21 on the left and hexadecimal values from 0 to f on the top. Most cells contain 'XX'. At address 20, the value '10' is highlighted with a red box, followed by '8f' at address 21.

Escreve na memória no endereço 201 o que tinha no acumulador (10).

Executando as próximas linhas:

LD 202

MM 10c

LD 203

MM 10d

The screenshot shows a memory editor interface with the following components:

- Memory List Table:**

CI	CO	OP
040	X	XXX
042	X	XXX
044	X	XXX
046	X	XXX
048	X	XXX
04a	X	XXX
04c	X	XXX
04e	X	XXX
050	X	XXX
052	X	XXX
054	X	XXX
056	X	XXX
058	X	XXX
05a	X	XXX
05c	X	XXX
05e	X	XXX
060	X	XXX
062	X	XXX

- Data Table:**

Endereco	Dado
XXX	XX
XXX	XX
XXX	XX
XXX	XX
XXX	XX
XXX	XX

- Input/Output Section:**

Entrada: LOAD

Endereço: 100 RUN

Saída: Next Step

Acomulador: 01 Reset

- Hex Dump:**

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
00	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
01	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
02	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
03	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
04	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
05	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
06	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
07	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
08	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
09	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
0a	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
0b	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
0c	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
0d	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
0e	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
0f	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
10	8f	00	92	01	82	02	91	0c	82	03	91	0d	8f	01	e0	04
11	82	01	52	00	92	01	11	2a	81	0d	42	00	91	0d	11	22
12	01	0c	81	0c	42	00	91	0c	01	0c	c1	00	XX	XX	XX	XX
13	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
14	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
15	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
16	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
17	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
18	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
19	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
1a	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
1b	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
1c	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
1d	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
1e	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
1f	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
20	01	10	8f	01	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
21	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Atualizo a instrução LDA INIC no endereço 10c.

Executando as próximas linhas:

LD f01

PD 004

The screenshot shows a debugger window with several panels. On the left, there is a table of memory locations (CI, CO, OP) and a table of registers (Endereco, Dado). Below these are input fields for 'Entrada', 'Endereço', 'Saída', and 'Acomulador', along with buttons for 'LOAD', 'RUN', 'Next Step', and 'Reset'. The 'Saída' field is highlighted with a red box and contains the value 83. The 'Acomulador' field is also highlighted with a red box and contains the value 53. On the right, there is a large memory dump showing hexadecimal values for addresses 00 to 21. The memory dump shows that the value 53 (0x35) is stored at address 00, and the value 83 (0x53) is stored at address 01.

CI	CO	OP
040	X	XXX
042	X	XXX
044	X	XXX
046	X	XXX
048	X	XXX
04a	X	XXX
04c	X	XXX
04e	X	XXX
050	X	XXX
052	X	XXX
054	X	XXX
056	X	XXX
058	X	XXX
05a	X	XXX
05c	X	XXX
05e	X	XXX
060	X	XXX
062	X	XXX

Endereco	Dado
XXX	XX
XXX	XX
XXX	XX
XXX	XX
XXX	XX
XXX	XX

Entrada: LOAD

Endereço: RUN

Saída: Next Step

Acomulador: Reset

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
00	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
01	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
02	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
03	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
04	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
05	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
06	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
07	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
08	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
09	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
0a	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
0b	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
0c	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
0d	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
0e	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
0f	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
10	8f	00	92	01	82	02	91	0c	82	03	91	0d	8f	01	e0	04
11	82	01	52	00	92	01	11	2a	81	0d	42	00	91	0d	11	22
12	01	0c	81	0c	42	00	91	0c	01	0c	c1	00	XX	XX	XX	XX
13	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
14	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
15	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
16	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
17	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
18	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
19	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
1a	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
1b	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
1c	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
1d	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
1e	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
1f	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
20	01	10	8f	01	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
21	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Mostra a primeira posição do vector de letras, neste caso "S", o qual é 53 em ASCII (1010011) e a saída do programa sempre se mostra em decimal por isso aparece 83.

Executando as próximas linhas:

LD 201
- 200
MM 201
JZ 12a

CI	CO	OP
040	X	XXX
042	X	XXX
044	X	XXX
046	X	XXX
048	X	XXX
04a	X	XXX
04c	X	XXX
04e	X	XXX
050	X	XXX
052	X	XXX
054	X	XXX
056	X	XXX
058	X	XXX
05a	X	XXX
05c	X	XXX
05e	X	XXX
060	X	XXX
062	X	XXX

Endereco	Dado
XXX	XX
XXX	XX
XXX	XX
XXX	XX
XXX	XX

Entrada: LOAD

Endereço: RUN

Saída: Next Step

Acomulador: Reset

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
00	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
01	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
02	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
03	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
04	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
05	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
06	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
07	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
08	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
09	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
0a	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
0b	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
0c	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
0d	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
0e	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
0f	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
10	8f	00	92	01	82	02	91	0c	82	03	91	0d	8f	01	e0	04
11	82	01	52	00	92	01	11	2a	81	0d	42	00	91	0d	11	22
12	01	0c	81	0c	42	00	91	0c	01	0c	c1	00	XX	XX	XX	XX
13	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
14	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
15	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
16	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
17	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
18	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
19	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
1a	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
1b	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
1c	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
1d	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
1e	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
1f	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
20	01	0f	8f	01	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
21	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX

Diminui o contador em um e guarda em memória no endereço 201 e se é 0 termina o loop.

E assim por diante executa tudo o código, mostrando o vector de letras e diminuido o contador até 0.

Executando as próximas linhas:

LD 10d
+ 200
MM 10d

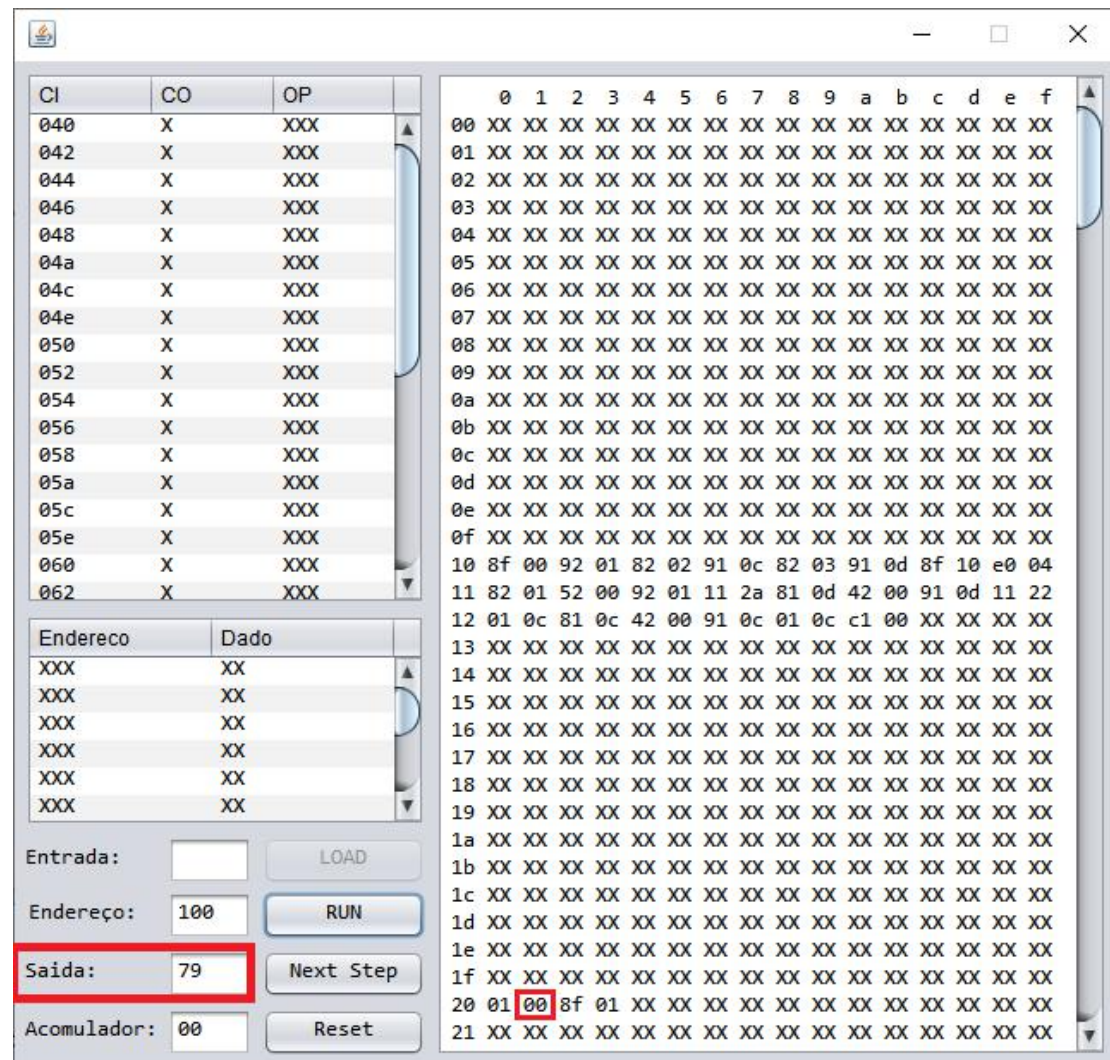
The screenshot shows a debugger window with the following components:

- Instruction List:** A table with columns CI, CO, and OP. It lists instructions from 040 to 062, all with CO 'X' and OP 'XXX'.
- Register/Variable Table:** A table with columns Endereco and Dado. It shows values 'XX' for addresses XXX, XXX, XXX, XXX, XXX, and XXX.
- Control Panel:** Includes fields for Entrada, Endereço (100), Saída (83), and Acomulador (02). Buttons for LOAD, RUN, Next Step, and Reset are also present.
- Memory Dump:** A large grid showing memory addresses from 00 to 21 and their corresponding hexadecimal values. The value '02' at address 10d is highlighted with a red box.

Atualiza a instrução LDA INIC no endereço 10d.

E assim por diante, executa tudo o código, mostrando o vector de letras, disminuido o contador até 0 e atualizando a instrução LDA INIC.

Finalmente executando tudo o código (click no botão RUN), na imagem baixo pode-se ver o resultado na saída (79), o qual é o numero em decimal da ultima letra ("O") de minha sequencia em ASCII 4f (1001111) e pode-se ver também o contador na memoria em 0.



Se quiser executar o programa dar [click aqui](#) e descarregar o executável (MVNQ1.jar).

Só tem que abrir o executável, dar click em LOAD e executar com RUN ou Next Step no caso Passo a Passo.

Segunda questão

Funcionalidade do program: Começa pelo dígito menos significativo (mais à direita), e vá preenchendo a célula correspondente do Código com o resto da divisão por 16 a soma dos valores dos dígitos já considerados até aquele ponto.

Ao final, o Código terá sido preenchido com oito valores, entre 0 e 15. Denote cada um deles na forma de um dígito hexadecimal.

NUSP	1	1	7	2	6	1	1	1
CODIGO	4	3	2	b	9	3	2	1

O programa recebe o vector NUSP em ASCII ou seja:

NUSP	1	1	7	2	6	1	1	1
ASCII	31	31	37	32	36	31	31	31

E com este vector ASCII gera o vector CODIGO.

Código objeto carregável

CI	CO	OP	ROTULO	COMENTARIO
411	LD	COMP	<i>START</i>	//comprimento do vector NUSP
413	MM	CONT		//inicia o contador do loop
415	LD	LD	<i>IN</i>	//constrói inicialmente a instrução LD
417	MM	LD0		//a ser diminuída a cada execução do loop
419	LD	LD+1		//a qual faz referencia ao vetor que contem
41b	MM	LD0+1		//o numero USP.
41d	LD	MM		//depois constrói-se a instrução MM
41f	MM	MM0		//a ser diminuída a cada execução do loop
421	LD	MM+1		//a qual faz referencia ao vetor que contem
423	MM	MM0+1		//o CODIGO.
425	LD	NUSP+7	<i>LD0</i>	//obter o numero USP
427	-	0a9		//transformação de ASCII a decimal
429	MM	NUSP+n		//salvar o resultado
42b	+	SUM		//realizar a soma com o acumulado
42d	MM	SUM		//salvar a soma
42f	-	BASE		//identificar se é maior que a base
431	JN	FUNC		//se é maior
433	LD	SUM		//carrega o resultado da soma
435	-	BASE		//resta a base
437	MM	SUM		//e salva o resultado
439	LD	SUM	<i>FUNC</i>	//se nao, carrega o resultado da soma
43b	MM	CODIGO+7	<i>MM0</i>	//e salva o resultado no vetor CODIGO
43d	LD	MM+1		//pega a posição atual no vetor CODIGO
43f	-	UM		//diminui a posição
441	MM	MM+1		//salva em memoria a siguiente posição
443	LD	LD+1		//pega a posição atual no vetor NUSP
445	-	UM		//diminui a posição
447	MM	LD+1		//salva em memoria a siguiente posição
449	LD	CONT		//pega o contador
44b	-	UM		//diminui um
44d	MM	CONT		//salva o contador
44f	JZ	FORA		//se o contador é zero, termina
451	JP	IN		//se não, começa com o siguiente número USP

453 HM START FORA //pára a máquina e volta ao início se for reacionada

200	145	MM	//91 em hexa	}	Instrução MM XXX
201	16	MM+1	//10 em hexa		
202	129	LD	//81 em hexa	}	Instrução LD XXX
203	8	LD+1	//08 em hexa		
204	48	0a9	//30 em hexa		
205	16	BASE	//10 em hexa		
206	1	UM			
207	0	CONT			
208	0	NUSP+n			
209	0	SUM	//acumulado		

100	8	COMP		
101	49	NUSP	//31 em hexa //1 em ASCII	
102	49		//31 em hexa //1 em ASCII	
103	55		//37 em hexa //7 em ASCII	
104	50		//32 em hexa //2 em ASCII	
105	54		//36 em hexa //6 em ASCII	
106	49		//31 em hexa //1 em ASCII	
107	49		//31 em hexa //1 em ASCII	
108	49		//31 em hexa //1 em ASCII	
109	0	CODIGO	//Debe resultar 4	
10a	0		//Debe resultar 3	
10b	0		//Debe resultar 2	
10c	0		//Debe resultar b	
10d	0		//Debe resultar 9	
10e	0		//Debe resultar 3	
10f	0		//Debe resultar 2	
110	0		//Debe resultar 1	

Uma vez tendo este código se compatibiliza com meu montador e fica assim:

Tabela do Programa:

IC	CO	OP
411	LD	100
413	MM	207
415	LD	202
417	MM	425
419	LD	203
41b	MM	426
41d	LD	200
41f	MM	43b
421	LD	201
423	MM	43c
425	LD	108
427	-	204
429	MM	208
42b	+	209
42d	MM	209
42f	-	205
431	JN	439
433	LD	209
435	-	205
437	MM	209
439	LD	209
43b	MM	110
43d	LD	201
43f	-	206
441	MM	201
443	LD	203
445	-	206
447	MM	203
449	LD	207
44b	-	206
44d	MM	207
44f	JZ	453
451	JP	415
453	HM	411

Tabela de dados:

200	145	MM	//91 em hexa
201	16	MM+1	//10 em hexa
202	129	LD	//81 em hexa
203	8	LD+1	//08 em hexa
204	48	0a9	//30 em hexa
205	16	BASE	//10 em hexa
206	1	UM	
207	0	CONT	
208	0	NUSP+n	
209	0	SUM	
100	8	COMP	
101	49	NUSP	//31 em hexa, 1 em ASCII
102	49		//31 em hexa, 1 em ASCII
103	55		//37 em hexa, 7 em ASCII
104	50		//32 em hexa, 2 em ASCII
105	54		//36 em hexa, 6 em ASCII
106	49		//31 em hexa, 1 em ASCII
107	49		//31 em hexa, 1 em ASCII
108	49		//31 em hexa, 1 em ASCII
109	0	CODIGO	//Debe resultar 4
10a	0		//Debe resultar 3
10b	0		//Debe resultar 2
10c	0		//Debe resultar b
10d	0		//Debe resultar 9
10e	0		//Debe resultar 3
10f	0		//Debe resultar 2
110	0		//Debe resultar 1

Implementação do código objeto na MVN

Primeramente escrevemos os dados e as instruções no programa:

The screenshot displays a memory editor window with the following components:

- Instruction Table:** A table with columns CI, CO, and OP. It lists instructions from address 411 to 433.
- Data Table:** A table with columns Endereco and Dado. It lists data from address 200 to 205.
- Memory Dump:** A grid showing memory addresses from 00 to 21 and their corresponding hexadecimal values (mostly XX).
- Control Fields:** Fields for Entrada, Endereço, Saída, and Acomulador, along with buttons for LOAD, Next Step, and Reset.

Red annotations highlight specific elements:

- A red box around the instruction table and a red arrow pointing to the memory dump at address 411 are labeled **tabela do programa**.
- A red box around the data table and a red arrow pointing to the memory dump at address 15 are labeled **tabela de dados**.
- A red box around the Endereço field (containing 411) and a red arrow pointing to the memory dump at address 1c are labeled **endereço inicial**.

Logo damos click em *LOAD* para montar tudo na memoria:

The screenshot shows a memory editor interface with the following components:

- Table 1 (CI, CO, OP):**

CI	CO	OP
040	X	XXX
042	X	XXX
044	X	XXX
046	X	XXX
048	X	XXX
04a	X	XXX
04c	X	XXX
04e	X	XXX
050	X	XXX
052	X	XXX
054	X	XXX
056	X	XXX
058	X	XXX
05a	X	XXX
05c	X	XXX
05e	X	XXX
060	X	XXX
062	X	XXX
- Table 2 (Endereco, Dado):**

Endereco	Dado
XXX	XX
XXX	XX
XXX	XX
XXX	XX
XXX	XX
XXX	XX
- Controls:**
 - Entrada:
 - Endereço:
 - Saida:
 - Acomulador:
- Memory Grid:** A grid showing memory addresses (00 to 21) and their contents (XX). The grid is organized into columns labeled 0 through f. The contents are mostly 'XX', but some cells are highlighted with red boxes:
 - Address 10: 08 31 31 37 32 36 31 31 31 00 00 00 00 00 00 00
 - Address 11: 00 XX XX XX XX XX XX XX XX XX XX XX XX XX XX
 - Address 20: 91 10 81 08 30 10 01 00 00 00 XX XX XX XX XX XX

Pode-se ver parte do código de maquina na memoria.

Executando as próximas linhas:

LD 100

MM 207

The screenshot shows a debugger window with the following components:

- Instruction List:** A table with columns CI, CO, and OP. It lists instructions from 040 to 062, all with CO 'X' and OP 'XXX'.
- Register/Variable List:** A table with columns Endereco and Dado. It shows six entries, all with 'XXX' in the Dado column.
- Input/Output Section:** Includes fields for 'Entrada:', 'Endereço:' (set to 411), 'Saída:', and 'Acomulador:' (set to 08). Buttons for 'LOAD', 'RUN', 'Next Step', and 'Reset' are present.
- Memory Dump:** A large grid showing memory addresses from 00 to 21. Most cells contain 'XX'. At address 20, the value is '91 10 81 08 30 10 01 08', where '08' is highlighted in yellow. At address 1f, the value is 'XX XX XX XX XX XX XX XX', where the first 'XX' is highlighted in yellow.

Escreve em memoria o comprimento do vetor NUSP (8).

Executando as próximas linhas:

LD 202
MM 425
LD 203
MM 426
LD 200
MM 43b
LD 201
MM 43c

The screenshot shows a 6502 assembly simulator interface. It features a memory window on the right displaying hexadecimal values for addresses 29 to 4b. A table on the left lists instructions with their CI, CO, and OP codes. Below this, a table shows the current state of registers and memory locations. At the bottom, there are input fields for 'Entrada:', 'Endereço:', 'Saída:', and 'Acomulador:', along with buttons for 'LOAD', 'RUN', 'Next Step', and 'Reset'.

CI	CO	OP
040	X	XXX
042	X	XXX
044	X	XXX
046	X	XXX
048	X	XXX
04a	X	XXX
04c	X	XXX
04e	X	XXX
050	X	XXX
052	X	XXX
054	X	XXX
056	X	XXX
058	X	XXX
05a	X	XXX
05c	X	XXX
05e	X	XXX
060	X	XXX
062	X	XXX

Endereco	Dado
XXX	XX
XXX	XX
XXX	XX
XXX	XX
XXX	XX
XXX	XX

Entrada: LOAD

Endereço: RUN

Saída: Next Step

Acomulador: Reset

Memory dump (addresses 29 to 4b):

```
29 XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX
2a XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX
2b XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX
2c XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX
2d XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX
2e XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX
2f XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX
30 XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX
31 XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX
32 XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX
33 XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX
34 XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX
35 XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX
36 XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX
37 XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX
38 XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX
39 XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX
3a XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX
3b XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX
3c XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX
3d XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX
3e XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX
3f XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX
40 XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX
41 XX 81 00 92 07 82 02 94 25 82 03 94 26 82 00 94
42 3b 82 01 94 3c 81 08 52 04 92 08 42 09 92 09 52
43 05 24 39 82 09 52 05 92 09 82 09 91 10 82 01 52
44 06 92 01 82 03 52 06 92 03 82 07 52 06 92 07 14
45 53 04 15 c4 11 XX XX XX XX XX XX XX XX XX XX
46 XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX
47 XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX
48 XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX
49 XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX
4a XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX
4b XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX
```

Inicia as instruções LD XXX e MM XXX.

Executando as próximas linhas:

```

425 LD 108
427 - 204
429 MM 208
42b + 209
42d MM 209
42f - 205
431 JN 439
433 LD 209
435 - 205
437 MM 209
439 LD 209
43b MM 110

```

The screenshot shows a debugger window with the following components:

- Disassembly Table:**

CI	CO	OP
040	X	XXX
042	X	XXX
044	X	XXX
046	X	XXX
048	X	XXX
04a	X	XXX
04c	X	XXX
04e	X	XXX
050	X	XXX
052	X	XXX
054	X	XXX
056	X	XXX
058	X	XXX
05a	X	XXX
05c	X	XXX
05e	X	XXX
060	X	XXX
062	X	XXX
- Register/Variable Table:**

Endereço	Dado
XXX	XX
XXX	XX
XXX	XX
XXX	XX
XXX	XX
XXX	XX
- Control Panel:**
 - Entrada:
 - Endereço: [LOAD] [RUN]
 - Saída: [Next Step]
 - Acomulador: [Reset]
- Hex Dump:** A table showing memory addresses (00 to 21) and their corresponding hexadecimal values. The value at address 11 is highlighted as 01.

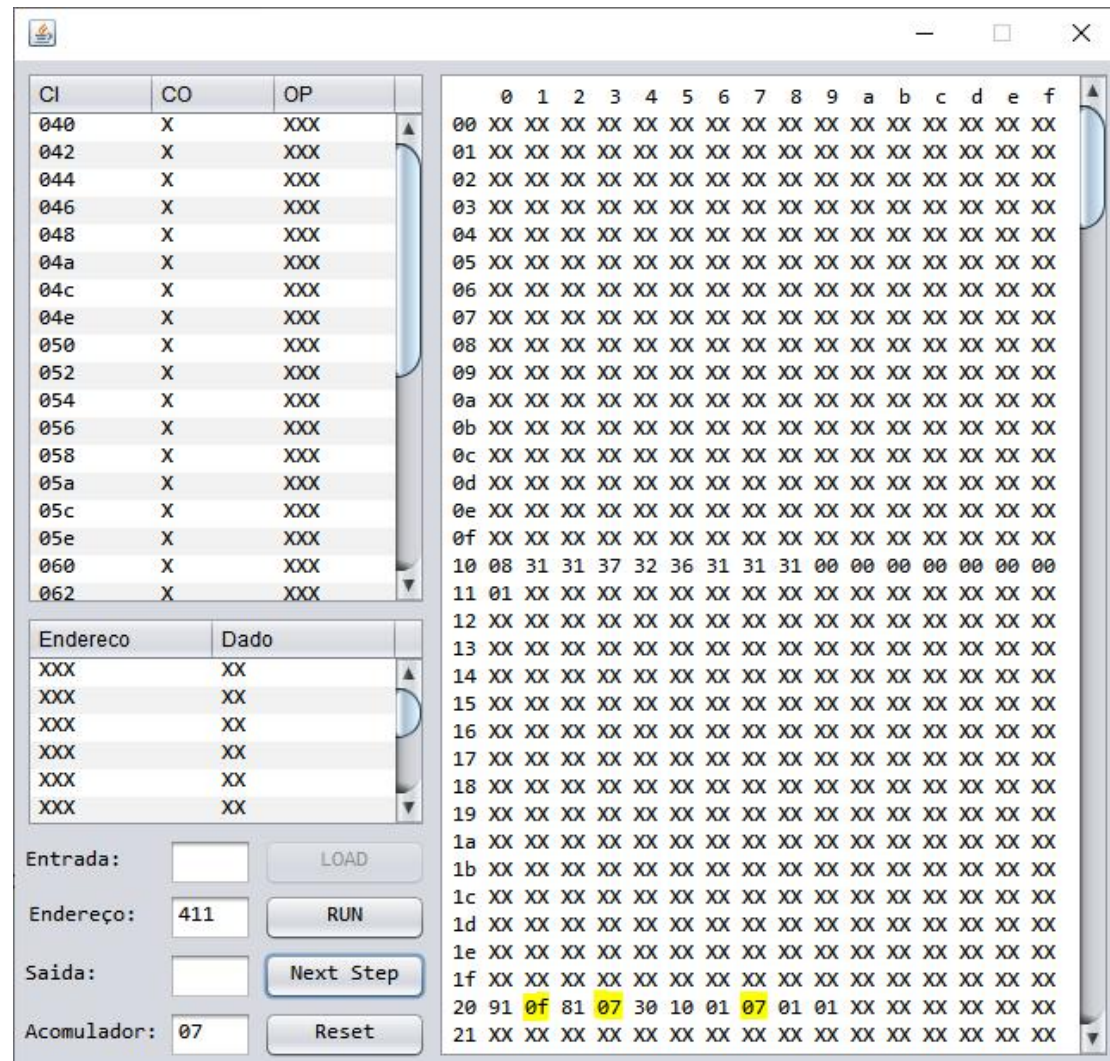
Pega o dígito menos significativo do vetor NUSP e começa a criar o vetor CODIGO.

Executando as próximas linhas:

```

43d LD 201
43f - 206
441 MM 201
443 LD 203
445 - 206
447 MM 203
449 LD 207
44b - 206
44d MM 207
44f JZ 453
451 JP 415

```

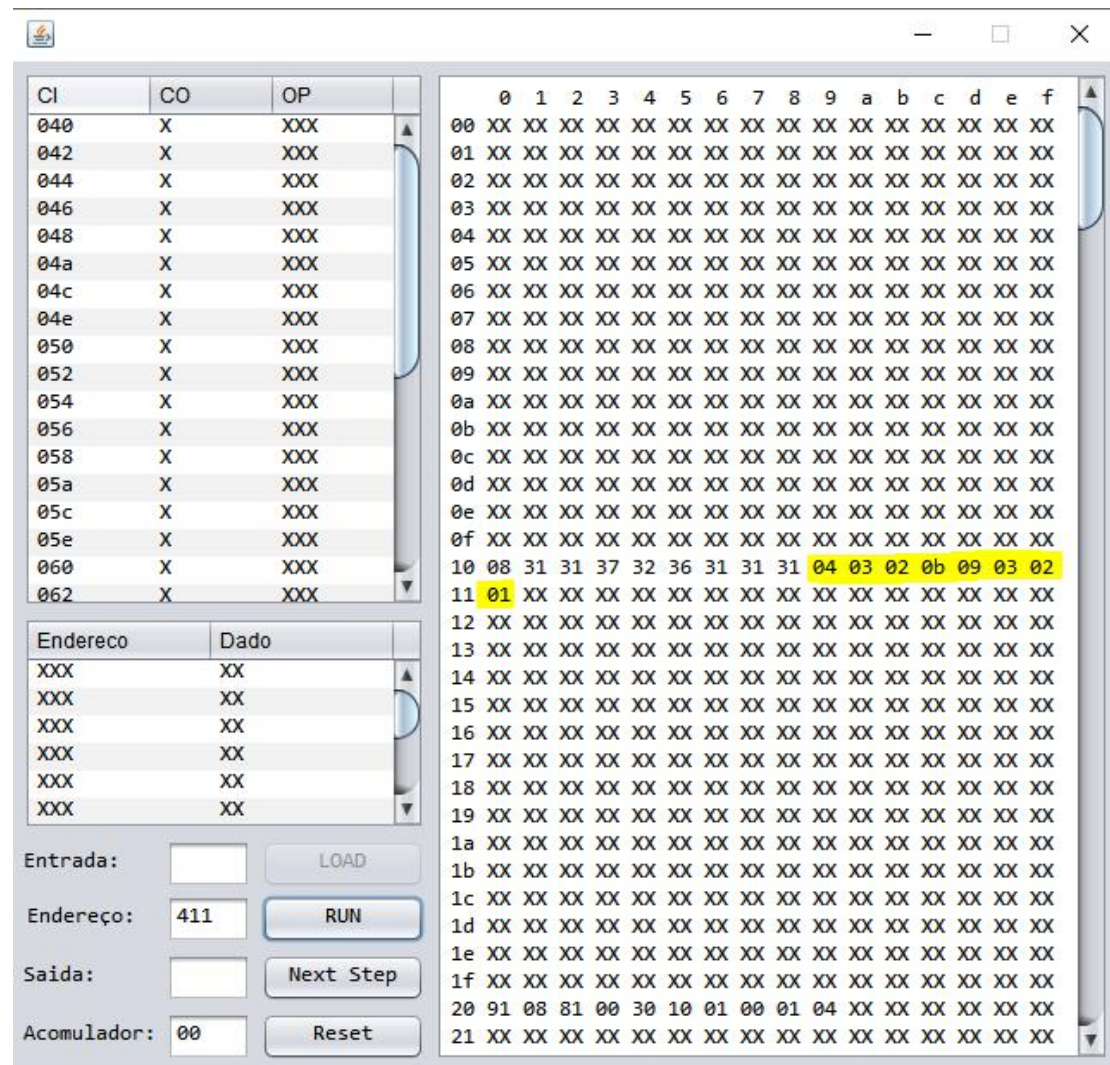


Diminui o contador, a posição a ser trabalhada do vetor NUSP e do vetor CODIGO, ou seja, as instruções LD XXX e MM XXX.

E assim por diante, executa tudo o código, diminuindo o contador, atualizando as instruções LD XXX e MM XXX e criando o vetor CODIGO até que o contador seja 0.

Finalmente executando o código tudo (com o botão RUN), na memória pode-se ver o vetor CODIGO esperado:

CODIGO	4	3	2	b	9	3	2	1
---------------	---	---	---	---	---	---	---	---



Se quiser executar o programa dar [click aqui](#) e descarregar o executável (MVNQ2.jar).

Só tem que abrir o executável, dar click em LOAD e executar com RUN ou Next Step no caso Passo a Passo.