Universidade Federal de Santa Catarina Centro Tecnológico - CTC
Departamento de Informática e Estatística - INE

Relatório atividade 3

Organização de computadores I

Prof.
Marcelo Daniel Berejuck

Alunos
Augusto de Hollanda Vieira Guerner (22102192)

Introdução

O presente documento tem por objetivo discorrer sobre a atividade 3 da matéria de org. Nessa atividade foi pedido que se escrevesse um programa em assembly equivalente ao programa em C que é dado no enunciado de cada questão (a e b). Resumidamente, a ideia de ambas questões é apenas criar dois laços para percorrer uma matriz, 16x16, e atribuir um dado valor a cada elemento dela.

Exercício 1)

Na primeira questão (a), é pedido que se faça uma interação na matriz de modo que ela seja preenchida linha por linha, indo da esquerda para a direita. Assim, a memória fica com uma sequência de 0 a 255. Fica mais evidente se olhar para a imagem 1. É interessante salientar o fato de que este primeiro exercício poderia ser feito com apenas um for e sem o uso de um contador externo, no entanto foi escolhido uma forma que fosse o mais fiel ao código em C, isto é, que não perdesse a essência dele.

Imagem 1.1 - Memória após a execução do programa a.

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0	1	2	3	4	5	6	7
0x10010020	8	9	10	11	12	13	14	15
0x10010040	16	17	18	19	20	21	22	23
0x10010060	24	25	26	27	28	29	30	31
0x10010080	32	33	34	35	36	37	38	39
0x100100a0	40	41	42	43	44	45	46	47
0x100100c0	48	49	50	51	52	53	54	55
0x100100e0	56	57	58	59	60	61	62	63
0x10010100	64	65	66	67	68	69	70	71
0x10010120	72	73	74	75	76	77	78	79
0x10010140	80	81	82	83	84	85	86	87
0x10010160	88	89	90	91	92	93	94	95
0x10010180	96	97	98	99	100	101	102	103
0x100101a0	104	105	106	107	108	109	110	111
0x100101c0	112	113	114	115	116	117	118	119
0x100101e0	120	121	122	123	124	125	126	127

0x10010000 (.data) Hexadecimal Addresses Hexadecimal Values ASCII

Fonte: Acervo do autor.

Imagem 1.2 - Memória após a execução do programa a.

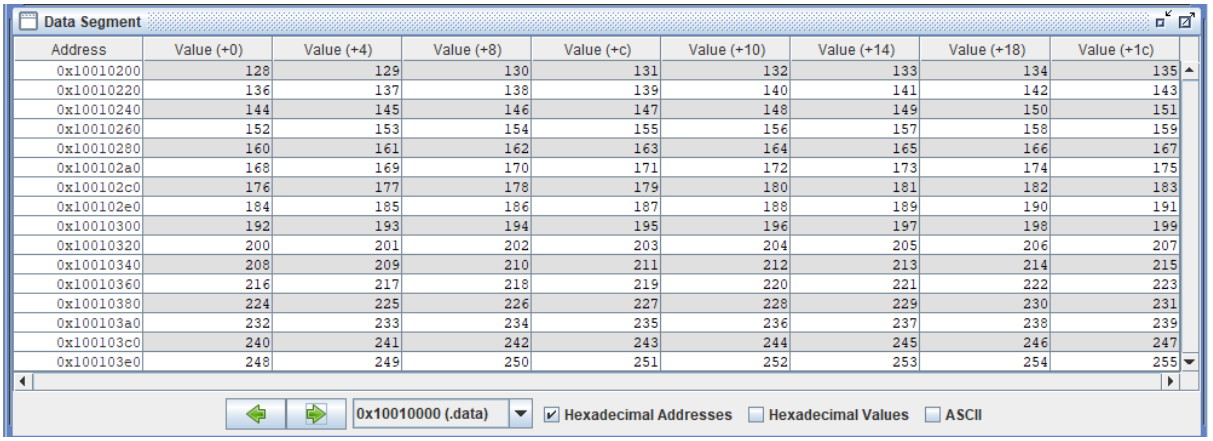
Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010100	64	65	66	67	68	69	70	71
0x10010120	72	73	74	75	76	77	78	79
0x10010140	80	81	82	83	84	85	86	87
0x10010160	88	89	90	91	92	93	94	95
0x10010180	96	97	98	99	100	101	102	103
0x100101a0	104	105	106	107	108	109	110	111
0x100101c0	112	113	114	115	116	117	118	119
0x100101e0	120	121	122	123	124	125	126	127
0x10010200	128	129	130	131	132	133	134	135
0x10010220	136	137	138	139	140	141	142	143
0x10010240	144	145	146	147	148	149	150	151
0x10010260	152	153	154	155	156	157	158	159
0x10010280	160	161	162	163	164	165	166	167
0x100102a0	168	169	170	171	172	173	174	175
0x100102c0	176	177	178	179	180	181	182	183
0x100102e0	184	185	186	187	188	189	190	191

0x10010000 (.data) Hexadecimal Addresses Hexadecimal Values ASCII

Fonte: Acervo do autor.

Imagem 1.3 - Memória após a execução do programa a.



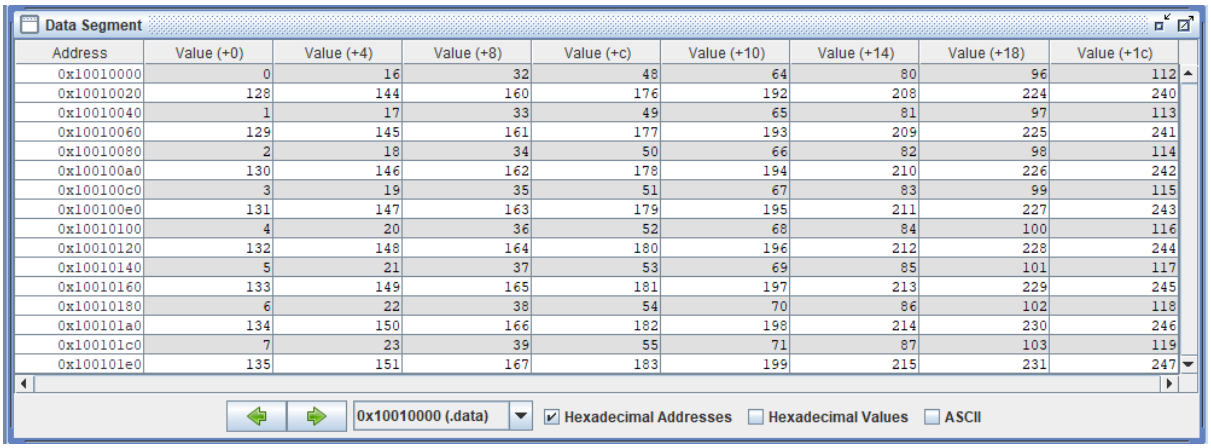
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010200	128	129	130	131	132	133	134	135
0x10010220	136	137	138	139	140	141	142	143
0x10010240	144	145	146	147	148	149	150	151
0x10010260	152	153	154	155	156	157	158	159
0x10010280	160	161	162	163	164	165	166	167
0x100102a0	168	169	170	171	172	173	174	175
0x100102c0	176	177	178	179	180	181	182	183
0x100102e0	184	185	186	187	188	189	190	191
0x10010300	192	193	194	195	196	197	198	199
0x10010320	200	201	202	203	204	205	206	207
0x10010340	208	209	210	211	212	213	214	215
0x10010360	216	217	218	219	220	221	222	223
0x10010380	224	225	226	227	228	229	230	231
0x100103a0	232	233	234	235	236	237	238	239
0x100103c0	240	241	242	243	244	245	246	247
0x100103e0	248	249	250	251	252	253	254	255

Fonte: Acervo do autor.

Exercício 2)

Na segunda questão (b), é exigido que se faça uma interação na matriz de forma que ela seja preenchida coluna por coluna, começando de cima e terminando embaixo. Dessa forma, a memória fica em primeiro momento desorganizada, no entanto há um padrão bem definido que é mais fácil visualizar por meio de imagem (imagem 2). Como é possível notar a sequência de números vai de 16 em 16 na memória, então se eu quero ver números em sequência eu tenho que pular 16 elementos para encontrar o próximo da sequência. Por exemplo, se começarmos na primeira posição da memória teremos o número 0; agora para encontrar o número seguinte, o 1, teremos que pular 16 elementos e assim por diante até chegarmos na última linha da matriz, que, neste caso, corresponderia ao número 15; chegando lá passamos agora para o segundo elemento da matriz, o 16, e faríamos o processo anterior novamente, até chegarmos ao número 255. No mais, pode-se observar que neste exercício poderia também se fazer com apenas um for, mas a implementação, além de perder a essência do código em C, ficaria tão grande quanto a de agora.

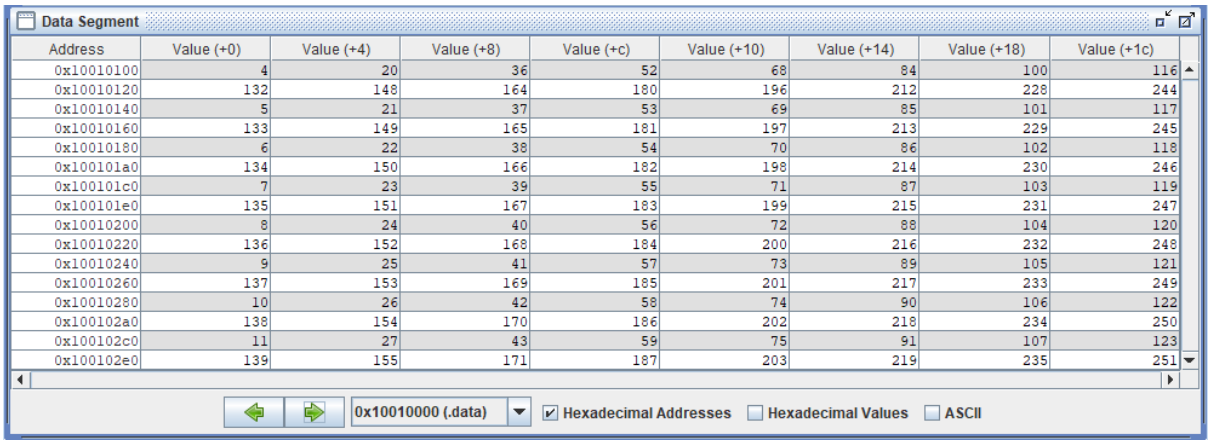
Imagem 2.1 - Memória após a execução do programa b.



Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0	16	32	48	64	80	96	112
0x10010020	128	144	160	176	192	208	224	240
0x10010040	1	17	33	49	65	81	97	113
0x10010060	129	145	161	177	193	209	225	241
0x10010080	2	18	34	50	66	82	98	114
0x100100a0	130	146	162	178	194	210	226	242
0x100100c0	3	19	35	51	67	83	99	115
0x100100e0	131	147	163	179	195	211	227	243
0x10010100	4	20	36	52	68	84	100	116
0x10010120	132	148	164	180	196	212	228	244
0x10010140	5	21	37	53	69	85	101	117
0x10010160	133	149	165	181	197	213	229	245
0x10010180	6	22	38	54	70	86	102	118
0x100101a0	134	150	166	182	198	214	230	246
0x100101c0	7	23	39	55	71	87	103	119
0x100101e0	135	151	167	183	199	215	231	247

Fonte: Acervo do autor.

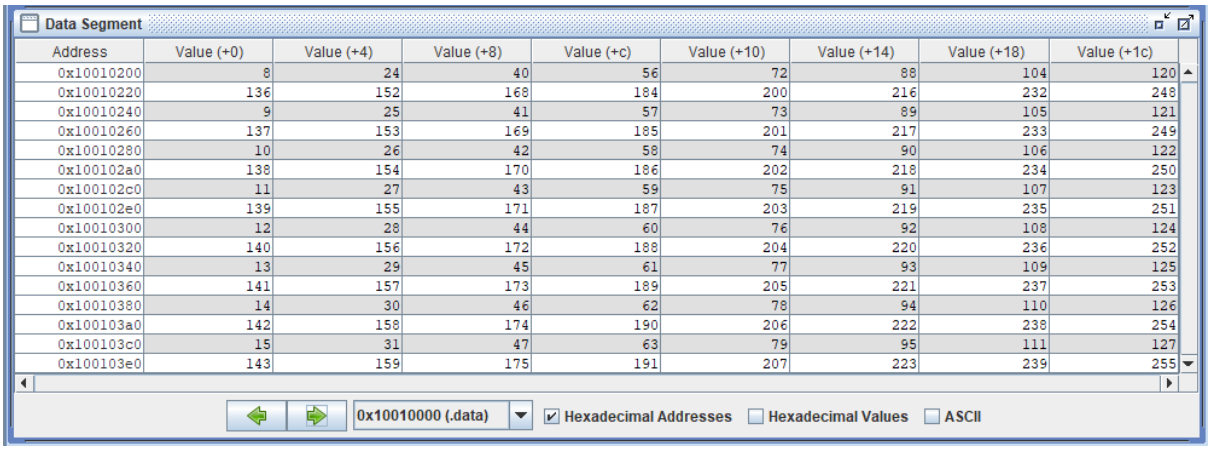
Imagem 2.2 - Memória após a execução do programa b.



Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010100	4	20	36	52	68	84	100	116
0x10010120	132	148	164	180	196	212	228	244
0x10010140	5	21	37	53	69	85	101	117
0x10010160	133	149	165	181	197	213	229	245
0x10010180	6	22	38	54	70	86	102	118
0x100101a0	134	150	166	182	198	214	230	246
0x100101c0	7	23	39	55	71	87	103	119
0x100101e0	135	151	167	183	199	215	231	247
0x10010200	8	24	40	56	72	88	104	120
0x10010220	136	152	168	184	200	216	232	248
0x10010240	9	25	41	57	73	89	105	121
0x10010260	137	153	169	185	201	217	233	249
0x10010280	10	26	42	58	74	90	106	122
0x100102a0	138	154	170	186	202	218	234	250
0x100102c0	11	27	43	59	75	91	107	123
0x100102e0	139	155	171	187	203	219	235	251

Fonte: Acervo do autor.

Imagem 2.3 - Memória após a execução do programa b.



Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010200	8	24	40	56	72	88	104	120
0x10010220	136	152	168	184	200	216	232	248
0x10010240	9	25	41	57	73	89	105	121
0x10010260	137	153	169	185	201	217	233	249
0x10010280	10	26	42	58	74	90	106	122
0x100102a0	138	154	170	186	202	218	234	250
0x100102c0	11	27	43	59	75	91	107	123
0x100102e0	139	155	171	187	203	219	235	251
0x10010300	12	28	44	60	76	92	108	124
0x10010320	140	156	172	188	204	220	236	252
0x10010340	13	29	45	61	77	93	109	125
0x10010360	141	157	173	189	205	221	237	253
0x10010380	14	30	46	62	78	94	110	126
0x100103a0	142	158	174	190	206	222	238	254
0x100103c0	15	31	47	63	79	95	111	127
0x100103e0	143	159	175	191	207	223	239	255

Fonte: Acervo do autor.

Conclusão

Tendo em vista o supracitado, percebe-se que a atividade, além de reforçar o entendimento de assembly, ela trabalhou também conceitos importantes como endereçamento. Vale salientar que ela na verdade aborda a memória de modo geral, colocando em prática o conhecimento teórico aprendido nas aulas.