

Disciplina: Fundamentos e Arquitetura de Computadores (FAC)

Professor: Tiago Alves

Trabalho 2 - Fundamentos de Arquitetura de Computadores

Alunos	Matrícula	Sistema Operacional
Djorkaeff Alexandre	16/0026822	macOS Sierra
Daniel Maike	16/0117003	Windows 10

O ambiente de desenvolvimento utilizado pelos dois alunos foi o MARS v4.5.

Instruções de uso:

Para simular os arquivos de resposta para as questões deste trabalho foi usado o aplicativo Mars versão 4.5 que pode ser baixado através de (<http://courses.missouristate.edu/kenvollmar/mars/>).

Após iniciar o aplicativo Mars, devemos abrir o arquivo de resposta da questão, na pasta zipada entregue neste trabalho temos um arquivo, com o nome "exp_mod.asm", estes arquivos são gerados pelo Mars e podem ser lidos pelo mesmo.

Após abrir um dos arquivos será lhe apresentado o código MIPS do mesmo.

The screenshot displays the MARS MIPS simulator interface. The main window shows assembly code for a program that prompts the user for a divisor, base, and exponent, then calculates the modular exponentiation result. The code includes comments in Portuguese. The right panel shows the MIPS registers, with \$a0, \$a1, and \$a2 containing values. The bottom panel shows the Mars Messages window, which displays the output of the program, including the prompt 'Informe o divisor (provavel primo): 13' and the result 'A exponencial modular 5 elevado a 3 (mod 13) eh 8'.

```
1      .data
2      Modulo: .asciiz "Informe o divisor (provavel primo): "
3      Base: .asciiz "Informe a base: "
4      Exponente: .asciiz "Informe o expoente: "
5      Primo: .asciiz "eh primo: "
6      Erro: .asciiz "o modulo nao eh primo: "
7      quebra_linha: .asciiz "\n"
8      resultado1: .asciiz "A exponencial modular "
9      resultado2: .asciiz " elevado a "
10     resultado3: .asciiz " (mod "
11     resultado4: .asciiz ")" eh "
12     Ponto: .asciiz "."
13
14     .text
15     le_inteiro: # Funcao para ler a base, o expoente e o possivel primo.
16
17     li $v0,4 # Carregando $v0, posso = carregar numero no registrador $v0, 4 = imprimir uma string.
18     la $a0,Base # Carrega o argumento em $a0.
19     syscall # Chamada de funcao.
20
21     li $v0,5 # Le um numero inteiro.
22     syscall # Chamada de funcao.
23
24     move $s1,$v0 # Move o conteudo de $v0 para o $s1 (BASE).
```

Register	Number	Value
\$zero	0	0
\$at	1	26550992
\$v0	2	4
\$v1	3	0
\$a0	4	26550152
\$a1	5	0
\$a2	6	0
\$a3	7	0
\$a4	8	0
\$a5	9	0
\$a6	10	0
\$a7	11	0
\$s0	12	0
\$s1	13	0
\$s2	14	0
\$s3	15	0
\$s4	16	0
\$s5	17	0
\$s6	18	0
\$s7	19	0
\$s8	20	0
\$s9	21	0
\$t0	22	0
\$t1	23	0
\$t2	24	0
\$t3	25	0
\$t4	26	0
\$t5	27	0
\$t6	28	265460224
\$t7	29	214747544
\$f0	30	0
\$f1	31	0
\$f2	32	0
\$f3	33	0
\$f4	34	0
\$f5	35	0
\$f6	36	0
\$f7	37	0
\$f8	38	0
\$f9	39	0
\$f10	40	0
\$f11	41	0
\$f12	42	0
\$f13	43	0
\$f14	44	0
\$f15	45	0
\$f16	46	0
\$f17	47	0
\$f18	48	0
\$f19	49	0
\$f20	50	0
\$f21	51	0
\$f22	52	0
\$f23	53	0
\$f24	54	0
\$f25	55	0
\$f26	56	0
\$f27	57	0
\$f28	58	0
\$f29	59	0
\$f30	60	0
\$f31	61	0
\$f32	62	0
\$f33	63	0
\$f34	64	0
\$f35	65	0
\$f36	66	0
\$f37	67	0
\$f38	68	0
\$f39	69	0
\$f40	70	0
\$f41	71	0
\$f42	72	0
\$f43	73	0
\$f44	74	0
\$f45	75	0
\$f46	76	0
\$f47	77	0
\$f48	78	0
\$f49	79	0
\$f50	80	0
\$f51	81	0
\$f52	82	0
\$f53	83	0
\$f54	84	0
\$f55	85	0
\$f56	86	0
\$f57	87	0
\$f58	88	0
\$f59	89	0
\$f60	90	0
\$f61	91	0
\$f62	92	0
\$f63	93	0
\$f64	94	0
\$f65	95	0
\$f66	96	0
\$f67	97	0
\$f68	98	0
\$f69	99	0
\$f70	100	0
\$f71	101	0
\$f72	102	0
\$f73	103	0
\$f74	104	0
\$f75	105	0
\$f76	106	0
\$f77	107	0
\$f78	108	0
\$f79	109	0
\$f80	110	0
\$f81	111	0
\$f82	112	0
\$f83	113	0
\$f84	114	0
\$f85	115	0
\$f86	116	0
\$f87	117	0
\$f88	118	0
\$f89	119	0
\$f90	120	0
\$f91	121	0
\$f92	122	0
\$f93	123	0
\$f94	124	0
\$f95	125	0
\$f96	126	0
\$f97	127	0
\$f98	128	0
\$f99	129	0
\$f100	130	0
\$f101	131	0
\$f102	132	0
\$f103	133	0
\$f104	134	0
\$f105	135	0
\$f106	136	0
\$f107	137	0
\$f108	138	0
\$f109	139	0
\$f110	140	0
\$f111	141	0
\$f112	142	0
\$f113	143	0
\$f114	144	0
\$f115	145	0
\$f116	146	0
\$f117	147	0
\$f118	148	0
\$f119	149	0
\$f120	150	0
\$f121	151	0
\$f122	152	0
\$f123	153	0
\$f124	154	0
\$f125	155	0
\$f126	156	0
\$f127	157	0
\$f128	158	0
\$f129	159	0
\$f130	160	0
\$f131	161	0
\$f132	162	0
\$f133	163	0
\$f134	164	0
\$f135	165	0
\$f136	166	0
\$f137	167	0
\$f138	168	0
\$f139	169	0
\$f140	170	0
\$f141	171	0
\$f142	172	0
\$f143	173	0
\$f144	174	0
\$f145	175	0
\$f146	176	0
\$f147	177	0
\$f148	178	0
\$f149	179	0
\$f150	180	0
\$f151	181	0
\$f152	182	0
\$f153	183	0
\$f154	184	0
\$f155	185	0
\$f156	186	0
\$f157	187	0
\$f158	188	0
\$f159	189	0
\$f160	190	0
\$f161	191	0
\$f162	192	0
\$f163	193	0
\$f164	194	0
\$f165	195	0
\$f166	196	0
\$f167	197	0
\$f168	198	0
\$f169	199	0
\$f170	200	0
\$f171	201	0
\$f172	202	0
\$f173	203	0
\$f174	204	0
\$f175	205	0
\$f176	206	0
\$f177	207	0
\$f178	208	0
\$f179	209	0
\$f180	210	0
\$f181	211	0
\$f182	212	0
\$f183	213	0
\$f184	214	0
\$f185	215	0
\$f186	216	0
\$f187	217	0
\$f188	218	0
\$f189	219	0
\$f190	220	0
\$f191	221	0
\$f192	222	0
\$f193	223	0
\$f194	224	0
\$f195	225	0
\$f196	226	0
\$f197	227	0
\$f198	228	0
\$f199	229	0
\$f200	230	0
\$f201	231	0
\$f202	232	0
\$f203	233	0
\$f204	234	0
\$f205	235	0
\$f206	236	0
\$f207	237	0
\$f208	238	0
\$f209	239	0
\$f210	240	0
\$f211	241	0
\$f212	242	0
\$f213	243	0
\$f214	244	0
\$f215	245	0
\$f216	246	0
\$f217	247	0
\$f218	248	0
\$f219	249	0
\$f220	250	0
\$f221	251	0
\$f222	252	0
\$f223	253	0
\$f224	254	0
\$f225	255	0
\$f226	256	0
\$f227	257	0
\$f228	258	0
\$f229	259	0
\$f230	260	0
\$f231	261	0
\$f232	262	0
\$f233	263	0
\$f234	264	0
\$f235	265	0
\$f236	266	0
\$f237	267	0
\$f238	268	0
\$f239	269	0
\$f240	270	0
\$f241	271	0
\$f242	272	0
\$f243	273	0
\$f244	274	0
\$f245	275	0
\$f246	276	0
\$f247	277	0
\$f248	278	0
\$f249	279	0
\$f250	280	0
\$f251	281	0
\$f252	282	0
\$f253	283	0
\$f254	284	0
\$f255	285	0
\$f256	286	0
\$f257	287	0
\$f258	288	0
\$f259	289	0
\$f260	290	0
\$f261	291	0
\$f262	292	0
\$f263	293	0
\$f264	294	0
\$f265	295	0
\$f266	296	0
\$f267	297	0
\$f268	298	0
\$f269	299	0
\$f270	300	0
\$f271	301	0
\$f272	302	0
\$f273	303	0
\$f274	304	0
\$f275	305	0
\$f276	306	0
\$f277	307	0
\$f278	308	0
\$f279	309	0
\$f280	310	0
\$f281	311	0
\$f282	312	0
\$f283	313	0
\$f284	314	0
\$f285	315	0
\$f286	316	0
\$f287	317	0
\$f288	318	0
\$f289	319	0
\$f290	320	0
\$f291	321	0
\$f292	322	0
\$f293	323	0
\$f294	324	0
\$f295	325	0
\$f296	326	0
\$f297	327	0
\$f298	328	0
\$f299	329	0
\$f300	330	0
\$f301	331	0
\$f302	332	0
\$f303	333	0
\$f304	334	0
\$f305	335	0
\$f306	336	0
\$f307	337	0
\$f308	338	0
\$f309	339	0
\$f310	340	0
\$f311	341	0
\$f312	342	0
\$f313	343	0
\$f314	344	0
\$f315	345	0
\$f316	346	0
\$f317	347	0
\$f318	348	0
\$f319	349	0
\$f320	350	0
\$f321	351	0
\$f322	352	0
\$f323	353	0
\$f324	354	0
\$f325	355	0
\$f326	356	0
\$f327	357	0
\$f328	358	0
\$f329	359	0
\$f330	360	0
\$f331	361	0
\$f332	362	0
\$f333	363	0
\$f334	364	0
\$f335	365	0
\$f336	366	0
\$f337	367	0
\$f338	368	0
\$f339	369	0
\$f340	370	0
\$f341	371	0
\$f342	372	0
\$f343	373	0
\$f344	374	0
\$f345	375	0
\$f346	376	0
\$f347	377	0
\$f348	378	0
\$f349	379	0
\$f350	380	0
\$f351	381	0
\$f352	382	0
\$f353	383	0
\$f354	384	0
\$f355	385	0
\$f356	386	0
\$f357	387	0
\$f358	388	0
\$f359	389	0
\$f360	390	0
\$f361	391	0
\$f362	392	0
\$f363	393	0
\$f364	394	0
\$f365	395	0
\$f366	396	0
\$f367	397	0
\$f368	398	0
\$f369	399	0
\$f370	400	0
\$f371	401	0
\$f372	402	0
\$f373	403	0
\$f374	404	0
\$f375	405	0
\$f376	406	0
\$f377	407	0
\$f378	408	0
\$f379	409	0
\$f380	410	0
\$f381	411	0
\$f382	412	0
\$f383	413	0
\$f384	414	0
\$f385	415	0
\$f386	416	0
\$f387	417	0
\$f388	418	0
\$f389	419	0
\$f390	420	0
\$f391	421	0
\$f392	422	0
\$f393	423	0
\$f394	424	0</

Figura 1 - Abertura de arquivo .asm no simulador Mars v4.5 código "*exp_mod.asm*"

Após aberto o arquivo `.asm` devemos montá-lo (simulação), para isso selecionamos no menu superior a opção `Run > Assemble`.

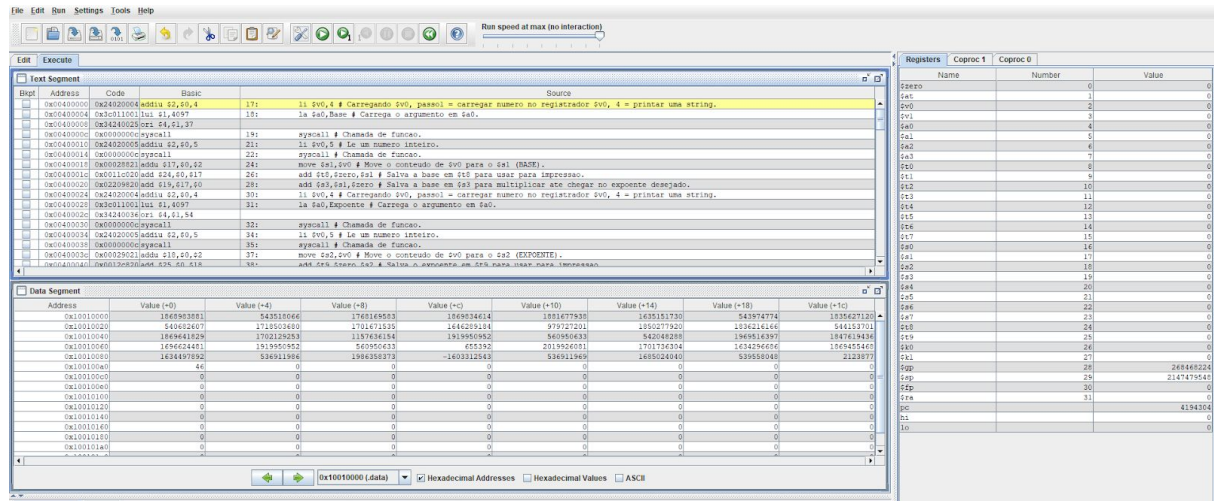


Figura 2 - Lista de comandos executados com o arquivo presente em *Text Segment*

Após clicar na opção Assemble presente na aba Run do menu superior, será apresentada a tela de Text Segment onde estarão presentes as instruções que serão realizadas por este arquivo sobre a memória simulada.



Deve-se clicar no botão indicado ao lado dentro do aplicativo Mars para que as instruções sejam simuladas.

Figura 3 - Saídas produzidas nos registradores pelo código

As entradas para as saídas representadas na figura 3 foram 5, 4 e 2.

Após clicar no botão que simula as instruções são geradas saídas nos registradores, os registradores.

Funcionamento do código:

O código primeiramente irá verificar se o divisor é primo para poder fazer o módulo. Se for igual a 1 ele irá imprimir o erro diretamente e se for igual a 2 ele irá direto para o cálculo da exponenciação modular. Para verificar se é primo, o código irá tirar a raiz quadrada do divisor (possível

primo) inserido pelo usuário do programa, se a raiz quadrada dele não for exata, ele irá pegar o número inteiro após o número com casas decimais (por exemplo: raiz quadrada de 6 ele irá pegar o 3), isso irá otimizar o código para que não precise percorrer todo o número digitado (faz com que possamos colocar números de até 31 bits, sendo o último o 2147483647 que é o último primo de 31 bits).

Ele irá percorrer a raiz quadrada do número inserido pelo usuário e fará divisões e quando o resto for igual a zero, ele irá incrementar um registrador utilizado para isso, após percorrer o número todo, se o registrador que incrementa quando o resto é igual a zero for maior que 1 (mais de um número teve resto igual a zero), ele irá imprimir o erro, caso contrário ele irá continuar para o cálculo da exponenciação modular.

Para tirar a exponenciação modular sem que estoure os bits, o código irá verificar se a base é maior que o número primo, se for ele irá calcular o módulo da base e irá voltar para o `calc_exp`, se não for ele irá verificar se o expoente é par, se for par ele irá dividir por 2 o expoente e irá elevar ao quadrado a base, reduzindo o expoente, caso o expoente seja ímpar ele irá subtrair 1 do expoente e irá pegar a base e multiplicar em um registrador temporário para guardá-lo para poder dividir o expoente por 2 e elevar a base ao quadrado, ele irá fazer esse loop até que o expoente seja igual a 1, com isso irá multiplicar o temporário e o restante da base e fazer o módulo final.

Limitações conhecidas:

Os números deverão ter 31 bits, entre 0 e 2.147.483.647, tanto o possível primo quanto a base elevada ao expoente, com isso o programa rodará corretamente seus resultados.