

Trabalho Prático 2 - Ligador

Jéssica D. A. Borges - jess.a.borges@gmail.com

Dezembro, 2015

I. INTRODUÇÃO

Foi proposto neste trabalho a implementação de um editor de ligação. O editor de ligação recebe os programas intermediários e o programa principal e os monta em um único programa executável. Os detalhes da entrada serão especificados na seção II. Os programas intermediários serão as saídas geradas pelo montador, que teve que ser modificado para mostrar informações adicionais, como detalhado na seção III

II. FORMATO DA CHAMADA AO LIGADOR

A chamada ao ligador deve fornecer os nomes dos arquivos intermediários dos módulos e do programa principal. Também deve fornecer o nome do arquivo de saída, onde estará o programa final executável. A chamada segue o padrão abaixo:

```
./bin/ligador tst/modulo1.mmh ... tst/moduloN.mmh -m tst/main.mmh -o tst/programa.mh
```

O formato dos programas de entrada serão especificados na seção III, pois serão o mesmo padrão da saída do montador modificado para este trabalho.

III. MONTADOR

Foi necessário fazer algumas modificações no montador implementado no trabalho prático 2. A primeira alteração é deixar labels não conhecidos sem tradução. Além disso, foi preciso deixar algumas informações no arquivo final. No início do arquivo, limitada por chaves, foi gravada a lista de símbolos do programa de entrada. Dentro das chaves, há um label por linha, cujas informações disponíveis são seu nome e seu endereço, separados por um espaço. Os símbolos desconhecidos tem seu endereço indicado com o caractere #. Após a tabela de símbolos, delimitado por colchetes, está o valor do *program counter* final, ou seja, o número de posições de memória aquele programa irá ocupar na memória da máquina virtual.

A chamada do montador sofre apenas uma pequena alteração em relação ao TP2. Para gerar a entrada do ligador, é necessário incluir '-L' ao final da chamada. Um exemplo de chamada está especificado abaixo:

```
./bin/montador tst/modulo.amh tst/modulo.mmh -L
```

IV. O PROCESSO DE LIGAÇÃO

O ligador primeiramente lê o início de todos os arquivos de entrada e salva as tabelas de símbolos e o número de posições que cada um ocupa na memória da máquina virtual. O ligador, então, manipula a tabela de símbolos do primeiro módulo, somando aos endereços dos símbolos o número de posições que o programa principal ocupa na memória. Soma-se o número de posições do programa principal ao número de posições do

primeiro módulo, em uma variável *currentPC*. Se houver outros módulos, o ligador repete o processo para cada módulo, somando agora o *currentPC*.

Terminado o processo descrito, o ligador percorre todas as tabelas de símbolos, procurando símbolos indefinidos. Os endereços desses símbolos são atualizados, considerando todas as outras tabelas de símbolos. Os símbolos desconhecidos todos atualizados, tabelas estão completas. O ligador, então, lê o programa principal e o escreve no arquivo de saída todo o código já traduzido. O código não traduzido são símbolos indefinidos, os quais são substituídos, no arquivo de saída, por seus endereços, disponíveis nas tabelas. O ligador repete o processo para cada ligador, seguindo a ordem da chamada.

V. ESTRUTURAS BÁSICAS DO LIGADOR

Module

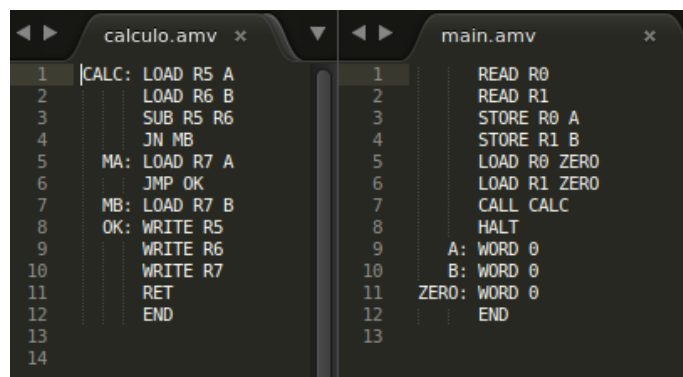
Esta estrutura contém a tabela de símbolos, o nome do arquivo que contém o código e o número de espaços o módulo ou programa principal irá ocupar na memória da máquina principal.

Linker

O *Linker* guarda o *main* e os módulos do programa, ambos de tipo *Module*, além do nome do arquivo de saída, onde serão combinados os módulos e o *main* em um programa executável.

VI. TESTES

O primeiro teste realizado foi o teste presente na documentação, cujos arquivos *calculo.amv* e *main.amv* estão disponíveis na pasta *tst*. Para este teste, serão mostrados os arquivos intermediários, para se ter uma ideia mais clara do formato de entrada do ligador. Porém, nos próximos testes, serão mostrados nesta documentação apenas os arquivos de entrada do montador e de saída do ligador. Os arquivos intermediários, porém, estarão disponíveis na pasta *tst*, com mesmo nome, porém final ".mmv".

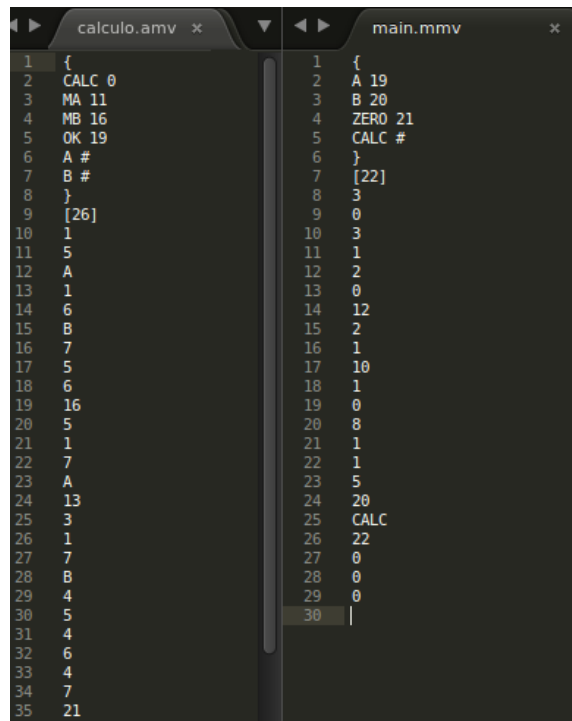


```
calculo.amv
1 |CALC: LOAD R5 A
2 |   LOAD R6 B
3 |   SUB R5 R6
4 |   JN MB
5 |   MA: LOAD R7 A
6 |   JMP OK
7 |   MB: LOAD R7 B
8 |   OK: WRITE R5
9 |   WRITE R6
10 |  WRITE R7
11 |   RET
12 |   END
13
14

main.amv
1 |   READ R0
2 |   READ R1
3 |   STORE R0 A
4 |   STORE R1 B
5 |   LOAD R0 ZERO
6 |   LOAD R1 ZERO
7 |   CALL CALC
8 |   HALT
9 |   A: WORD 0
10 |  B: WORD 0
11 |  ZERO: WORD 0
12 |   END
13
```

Figure 1: Teste 1 - entrada do montador

O segundo teste é uma calculadora. O teste está disponível na pasta *tst/calculadora*. O programa recebe três números *a*, *b* e *c*. O primeiro número indica a operação a ser realizada. Para *a* = 1, a operação é soma, disponível no arquivo *soma.amv*. Para *a* = 2, a operação a ser realizada é a subtração, no arquivo *subtrai.amv*. Para *a* = 3, multiplica-se, usando o módulo em *multiplica.amv*. Para *a* = 4, divide-se, usando o módulo *divisao.amv*. Finalmente, para *a* = 5, a operação a ser realizada é a potenciação, através do módulo *potencia.amv*. Os números *b* e *c* são os operandos da calculadora. O programa principal pode ser encontrado



```
calculo.amv
1 {
2  CALC 0
3  MA 11
4  MB 16
5  OK 19
6  A #
7  B #
8  }
9  [26]
10 1
11 5
12 A
13 1
14 6
15 B
16 7
17 5
18 6
19 16
20 5
21 1
22 7
23 A
24 13
25 3
26 1
27 7
28 B
29 4
30 5
31 4
32 6
33 4
34 7
35 21

main.mmv
1 {
2  A 19
3  B 20
4  ZERO 21
5  CALC #
6  }
7  [22]
8  3
9  0
10 3
11 1
12 2
13 0
14 12
15 2
16 1
17 10
18 1
19 0
20 8
21 1
22 1
23 5
24 20
25 CALC
26 22
27 0
28 0
29 0
30 |
```

Figure 2: *Teste 1 - entrada do ligador*

em main.amv e também visto logo abaixo. Para ficarem mais claras as ligações entre os diversos módulos, também pode-se ver as tabelas completas. A saída era muito extensa e, por isso, não está nesta documentação, mas pode ser encontrada no arquivo programa.mv, dentro da pasta tst/calculadora.

calculo.n	3	programa.mv
1	3	1 19
2	0	2 1
3	3	3 6
4	1	4 20
5	2	5 7
6	0	6 5
7	12	7 6
8	2	8 16
9	1	9 5
10	10	10 1
11	1	11 7
12	0	12 19
13	8	13 13
14	1	14 3
15	1	15 1
16	5	16 7
17	20	17 20
18	22	18 4
19	22	19 5
20	0	20 4
21	0	21 6
22	0	22 4
23	1	23 7
24	5	24 21

Figure 3: Teste 1 - saída do ligador

<pre> ;Programa principal: recebe a, b e c ; a indica a operação ; b e c são os operandos READ R11 ;a READ R2 ;b READ R3 ;c STORE R2 OPB STORE R3 OPC LOAD R7 AD SUB R11 R7 JZ SOMA ADD R11 R7 LOAD R7 SU SUB R11 R7 JZ SUBTRAI ADD R11 R7 LOAD R7 MU SUB R11 R7 JZ MULTIPLICA </pre>	<pre> ADD R11 R7 LOAD R7 DI SUB R11 R7 JZ DIVIDE ADD R11 R7 LOAD R7 PO SUB R11 R7 JZ POTENCIA X: WORD 0 Y: WORD 1 AD: WORD 1 SU: WORD 2 MU: WORD 3 DI: WORD 4 PO: WORD 5 OPB: WORD 0 OPC: WORD 0 </pre>
--	--

Figure 4: Teste 2 - entrada do montador

{	{	
X 64	MULTIPLICA 165	
Y 65	VERIFICAR3 190	
AD 66	CONTINUA 197	
SU 67	R2NEGATIVO 202	
MU 68	R2POSITIVO 209	
DI 69	R3NEGATIVO 214	
PO 70	R3POSITIVO 221	
OPB 71	MULT 226	
OPC 72	M 229	
SOMA 282	VSINAL 239	
SUBTRAI 294	R2NNEG 246	
MULTIPLICA 165	R3NNEG 253	
DIVIDE 73	A 254	
POTENCIA 207	B 255	
}	OPB 71	
{	OPC 72	
DIVIDE 73	}	
1 104	{	
2 114	POTENCIA 207	
DIV 123	POW 224	
FIMD 135	PNZERO 235	
RESTO 136	FIMP 250	{
SINAIS 144	PZERO 251	SOMA 282
OP2 153	MULT 256	OPB 71
FIMS 162	MNZERO 269	OPC 72
A 163	FIMM 279	}
B 164	A 280	{
OPB 71	B 281	SUBTRAI 294
OPC 72	OPB 71	OPB 71
}	OPC 72	OPC 72
	}	}

Figure 5: *Teste 2 - tabelas de símbolos*