1º Trabalho Prático
CIC 116432 – Software Básico
Prof. Bruno Macchiavello
2º Semestre de 2017

1 Introdução

O trabalho consiste em implementar em C/C++ um método de tradução de uma linguagem de montagem simples para uma representação de código objeto. Este trabalho foca na elaboração de um ligador e carregador.

2 Objetivo

Fixar o funcionamento de um processo de tradução. Especificamente as de carregamento e ligação.

3 Especificação

3.1 Montador

Modifique o seu trabalho anterior para realizar um montador que consegui trabalhar com códigos em módulos. Para isso, duas novas diretivas são necessárias: BEGIN e END, conforme a Tabela 1. O montador então deve receber de 1 a 3 programas de entrada, por argumento, um seguido do outro (ex.: ./montador prog1.asm prog2.asm prog3.asm). Se um único programa foi colocado então o mesmo NÃO deve ter as diretivas BEGIN e END. Se 2 ou 3 programas são definidos pelo usuário como entrada então as diretivas BEGIN e END são obrigatórias. Este então é a único teste de erro que o montador deve fazer, todos os testes de erros, assim como a parte de MACROS, do trabalho anterior NÃO serão avaliados neste trabalho. A utilização de BEGIN e END deve seguir os slides de sala de aula.

O montador deve então dar como saída de 1 a 3 arquivos objetos. Em todos eles deve existir um cabeçalho com pelo menos as seguintes informações: Nome do programa (pode ser o mesmo nome do arquivo), tamanho do código, e informação de realocação. A informação de realocação pode ser feita por lista de endereços ou mapa de bits. Exemplo:

Arquivo de OBJETO de saída:

```
H: PROG1
H: 12
H: 010010101010
T: 12 14 15 02 5 15 12 1 6 9 4 2
```

No exemplo anterior o cabeçalho é identificado pela letra H, enquanto a letra T indica a parte de texto (código). Além disso, se necessário o cabeçalho deve também incluir a TABELA DE USO e TABELA DE DEFINIÇÕES. O formato da inclusão dessa tabela é livre, ou seja, o grupo pode escolher como incluir ela no arquivo. Porém, deve estar claramente identificada pelo rótulo TU e TD especificamente. Ou seja, deve ter uma ou mais linhas com esse rótulo, como a informação na tabela é armazenada pode ser decidido pelo grupo. Os arquivos de saída devem ser .o

3.2 Ligador

Realizar um programa ligador.c que recebe de 1 a 3 arquivos de objeto. Os arquivos de objeto de entrada **serão** a saída do montador da parte anterior do seu própio montador. A ordem de entrada dos programas define também a ordem que eles devem ser ligados. O ligador deve então realizar o processo de ligação e dar como saída um único arquivo, onde as informações de cabeçalho devem ser unicamente: Nome do programa (pode ser o mesmo nome do primeiro arquivo objeto), tamanho do código, e informação de realocação. O arquivo de saída não deve ter extensão.

3.3 Carregador Realocador

Realizar um programa carregador.c que recebe como entrada um arquivo de saída do ligador da seção anterior e depois uma sequência de números. Exemplo: ./carregador prog1 5 12 3 7 4. Inicialmente, este programa deve executar o código fazendo uma simulação (deve simular o código em funcionamento). Após simular o funcionamento do programa, o carregador deve gerar um arquivo de saída de imagem de memória. Para isso são utilizados os números que foram inseridos junto com o nome do arquivo pelo usuário. Seguindo o exemplo anterior (./carregador prog1 3 12 3 7 100 150 280), o primeiro número indica a quantidade de CHUNKS de memória disponíveis neste momento, no caso 3. Os próximos números indicam o tamanho em bytes de cada um desses chunks (no exemplo 12, 3 e 7 bytes). Os últimos números indicam o endereço inicial de cada chunk. O carregador então deve verificar se um desses chunks é suficiente para suportar o programa inteiro, se não deve verificar se é possível então dividir em programa em diferentes chunks. Caso, não tenha como alocar o programa, mesmo dividindo, o carregador deve indicar uma mensagem de: “OUT OF MEMORY - YOUR PROGRAM WILL NOT BE LOADED”. Caso contrário, o carregador deve modificar a informação relativa para os endereços corretos e colocar o código final (sem cabeçalho)

no arquivo de saída. O arquivo de saída deve ter extensão .im, e deve ter unicamente os códigos e valores que estariam emc ada endereço de memória.

A forma de entrega é pelo Moodle. O trabalho pode ser feito em dupla (não individualmente).

Tabela 1: Intruções e diretivas.

Instruções				
Mnemônico	Operandos	Código	Tamanho	Descrição
ADD	1	1	2	ACC <- ACC + MEM[OP]
SUB	1	2	2	ACC <- ACC - MEM[OP]
MULT	1	3	2	ACC <- ACC * MEM[OP]
DIV	1	4	2	ACC <- ACC / MEM[OP]
JMP	1	5	2	PC <- OP
JMPN	1	6	2	Se ACC < 0, PC <- OP
JMPP	1	7	2	Se ACC > 0, PC <- OP
JMPZ	1	8	2	Se ACC = 0, PC <- OP
COPY	2	9	3	MEM[OP2] <- MEM[OP1]
LOAD	1	10	2	ACC <- MEM[OP]
STORE	1	11	2	MEM[OP] <- ACC
INPUT	1	12	2	MEM[OP] <- STDIN
OUTPUT	1	13	2	STDOUT <- MEM[OP]
STOP	0	14	1	Encerrar execução.
Diretivas				
SECTION	1	-	0	Marcar início de seção de código (TEXT) ou dados (DATA).
SPACE	0/1	-	variável	Reservar 1 ou mais endereços de memória não-inicializada para armazenamento de uma palavra.
CONST	1	-	1	Reservar memória para armazenamento de uma constante inteira de 16 *bits* em base decimal ou hexadecimal.
EQU	1	-	0	Cria um sinônimo textual para um símbolo
IF	1	-	0	Instrue o montador a incluir a **linha seguinte** do código somente se o valor do operando for 1
MACRO	0	-	0	Marcar início de suma MACRO. Sempre dentro da seção TEXT e antes do código principal
ENDMACRO	0	-	0	Marcar o fim de uma MACRO.
BEGIN	0	-	0	Inicio de um Módulo.
END	0	-	0	Fim de um Módulo.