2º Trabalho Prático
CIC 116432 – Software Básico
Prof. Bruno Macchiavello
1º Semestre de 2018

1 Objetivo

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

2 Especificação

2.1 Montador

Modifique o seu trabalho anterior para realizar um montador que consegui trabalhar com códigos em módulos. Para isso, as novas diretivas são necessárias: BEGIN, END, PUBLIC e EXTERN conforme a Tabela 1. O montador então deve receber de 1 a 2 programas de entrada, por argumento, um seguido do outro (ex.: ./montador prog1 prog2). Se um único programa foi colocado então o mesmo NÃO deve ter as diretivas BEGIN e END. Se 2 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, EQU e IF 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 2 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.

NOTAR QUE O FORMATO DE ARQUIVO É LIVRE LOGO PARA TESTAR O LIGADOR O MONTADOR DEVE DAR A SAÍDA CORRETA. SOMENTE SERÁ TESTADO O LIGADOR COM AS SAÍDAS DO SEU PRÓPRIO MONTADOR.

2.2 Ligador

Realizar um programa ligador.c que recebe de 1 a 2 arquivos de objeto. Os arquivos de objeto de entrada **serão** a saída do montador da parte anterior. 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 objeto, sim cabeçalho que deve poder ser utilizado nos simuladores como o Trabalho 1.

2.3 Entrada e Saída de DADOS em IA-32

Esta seção não é relacionada as seções anteriores e será avaliada separadamente. Faça um programa em IA-32 que represente uma calculadora que possa realizar as operações de SOMA, SUBSTRAÇÃO, MULTIPLICAÇÃO e DIVISÃO e MOD. O programa deve inicialmente perguntar para o usuário seu nome, depois dar uma mensagem de boas vindas ("Hóla, ¡nome do usuário¿, bem-vindo ao programa de CALC IA-32"). Depois deve mostrar um menú:

ESCOLHA UMA OPÇÃO:

- 1: SOMA
- 2: SUBSTRAÇÃO
- 3: MULTIPLICAÇÃO
- 4: DIVISÂO
- 5: MOD
- 6: SAIR

A opção 6 deve terminar o programa. Qualquer opção entre 1 a 5, deve pedir 2 argumentos assumindo que são números inteiros COM SINAL (podendo ser representados com até 32 bits). Deve mostrar o resultado final. Esperar o usuário digitar ENTER e novamente mostrar o mesmo menú. Para entrada e saída de dados o programa deve usar FUNÇÕES, uma função para ler números inteiros com sinal, uma função para escrever números inteiros com sinal, uma função para ler strings e uma função para escrever strings. O programa pode usar passagem de parâmetros pela pilha ou por registradores.

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.
BEGIN	0	-	0	Inicio de um Módulo.
END	0	-	0	Fim de um Módulo.
PUBLIC	0	-	0	Rótulo Público
EXTERN	0	-	0	Rótulo Externo