2º Trabalho Prático
CIC 116432 – Software Básico
Prof. Bruno Macchiavello
1º Semestre de 2015

1 Introdução

O trabalho consiste em duas partes: (*i*) implementar em C/C++ um método de tradução de uma linguagem de montagem simples para uma representação de código objeto e IA-32, (*ii*) implementar um programa em C/C++ um arquivo executável em formato ELF 32 bits.

2 Objetivo

Fixar o funcionamento de um processo de ligação e formato de arquivos.

3 Especificação

3.1 Tradutor

O programa tradutor (tradutor.c) deve receber um arquivo (arquivo.asm) como argumento. Este arquivo deve estar na linguagem Assembly hipotetica vista em sala de aula. Sendo que deve estar separadas em seções de dados e códigos. Esta linguagem é formada por um conjunto de instruções e diretivas segundo a tabela no final desta especificação. Note que as diretivas EQU e IF estão presentes. Os programas de entrada serão sempre de um único módulo. O tradutor deve ser capaz de:

- NÃO ser sensível ao caso, podendo aceitar instruções/diretivas/rótulos em maiúsculas e minúsculas.
- A seção de dados pode vir antes ou depois da seção de texto.
- A diretiva EQU sempre deve vir no inicio do programa.
- Desconsiderar tabulações e espaços desnecessários em qualquer lugar do código.

- A diretiva CONST deve aceitar declaração em hexadecimal também;
- Deve ser possível trabalhar com vetores
- Capacidade de aceitar comentários indicados pelo símbolo ";"
- O comando COPY deve utilizar uma vírgula entre os operandos (COPY A, B)
- Identificar pelo menos os seguintes erros (simplesmente indicando a linha do erro, não é necessário indicar tipo do erro):
  - declarações ausentes;
  - declarações repetidas;
  - diretivas ou instruções inválidas;
  - diretivas ou instruções na seção errada;
  - instruções com a quantidade de operando inválida;
  - tokens inválidos;
  - rótulos repetidos;
  - seção (TEXT) faltante;

O programa deve entregar duas saídas. A primeira um arquivo em formato texto (arquivo.s) que deve ser a tradução do programa de entrada em Assembly IA-32. As diretivas EQU e IF já devem ter sido avalidas antes da tradução.

Para ler e mostrar números, será necessário duas funções *LeerInteiro* e *EscreverInteiro*, respectivamente. Estão funções devem ser a tradução para INPUT and OUTPUT, respectivamente. As funções devem ser chamadas no Assembly mediante o comando CALL como visto em sala de aula. As mesmas devem ser desenvolvidas utilizando chamadas ao sistema. As funções devem estar prontas no seu programa utilizando o formato NASM, não é permitido o uso da bibleoteca "io.mac".

A função *LeerInteiro* deve ler vários caracteres do teclado, até o ENTER (0x0A) ser digitado. Para cada caracter assumir que é um número, transformar de ASCII para inteiro (subtrair 0x30) e fazer as operações aritméticas necessárias para criar um único número.

A função *EscreverInteiro* de formar similiar deve fazer as operações aritméticas para transformar o número numa sequência de strings, lembrando de adicionar 0x30 a cada digito. E no final pular de linha.

Observe que as seções de texto e dados da linguagem de montagem hipotética devem ser convertidas para o novo formato de forma a conservar o comportamento correto do programa.

A segunda saida deve ser um arquivo (arquivo.cod) que deve conter em formato texto o código máquina das instruções separadas por espaço sem nenhum tipo de formatação.

3.2 ELF32

Além dos duas saídas mencionadas na parte anterior, o programa deve retornar uma terceira saida que deve conter instruções em linguagem máquina, obtidas a partir da primeira parte do trabalho. Esta saída do programa (arquivo sem extensão) deve ser um arquivo executável em formato ELF32 capaz de ser executado em qualquer máquina INTEL 386 ou superior, rodando SO LINUX. Para isso recomenda-se o uso da biblioteca "libelf" para a criação do arquivo ELF32. Verificar no arquivo "test-elf.c" no Moodle o uso da biblioteca "libelf" para a criação do arquivo ELF32. O uso da biblioteca não é obrigatória o programa pode gerar os cabe¸calhos e estruturas necessárias sem utilizar a biblioteca.

Somente é necessário verificar os OPCODES das 14 instruções equivalentes em IA-32 do Assembly hipotético e das instruções utilizadas nas funções de ler e escrever inteiro e das instruções utilizadas nas sub-rotinas de I/O. Para verificar OPCODES http://www.mathemainzel.info/files/x86asmref.html#call ou verificar o manual da INTEL no Moodle. Lembre que as diretivas são avaliadas durante montagem/ligação e não geram código objeto.

4 Avaliação

O prazo de entrega do trabalho é 3 de Julho de 2015. A entrega consistirá em:

- A primeira parte do trabalho vale 8 pontos.
- A parte de ELF32 vale 2 pontos.

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

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	1	-	1	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