2º Trabalho Prático
CIC 116432 – Software Básico
Prof. Bruno Macchiavello
1º Semestre de 2014

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 ligador (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 apenas 14 instruções (Tabela no final).

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. Para ler e mostrar números, será necessário duas funções *LeerInteiro* e *EscreverInteiro*, 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.

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 em formato binário (arquivo.bin) que deve ser um sequência de bytes contendo os OPCODES do programa *arquivo.s*. Este arquivo deve conter unicamente os OPCODES sem nenhum tipo de formatação.

3.2 ELF32

Realizar um programa carregador (carregador.c) que recebe um arquivo binário (arquivo.bin). O arquivo binário deve conter instruções (Opcodes) da linguagem IA-32, obtidas a partir da primeira parte do trabalho. A saída do programa (arquivo) deve ser um arquivo executável em formato ELF32 capaz de ser executado em qualquer máquina INTEL 386 ou superior, rodando SO LINUX.

4 Avaliação

O prazo de entrega do trabalho é 23 de Junio de 2014. A entrega consistirá em:

- Código-fonte completo e comentado com instruções de compilação dos programas de tradução e simulação;
- Programas de exemplo que demonstrem o funcionamento correto dos programas de tradução e simulação.

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

5 Dicas

Verificar no arquivo "test-elf.c" o uso da biblioteca "libelf" para a criação do arquivo ELF32. O uso da biblioteca não é obrigatória o programa pode gerar os cabeçalhos 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. (http://www.mathemainzel.info/files/x86asmref.html#call)

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