2º Trabalho Prático
CIC 116432 – Software Básico
Prof. Bruno Macchiavello
1º Semestre de 2019

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 hipotética vista em sala de aula (com algumas modificações, a serem descritas posteriormente). Sendo que deve estar separadas em seções de dados e códigos. A seção de dados deve sempre vir no final. Não será avaliado detecção de erros léxicos, semânticos ou sintáticos. Ou seja, os programas não vao conter erros, mas o analisador sintático do trabalho anterior é util para a implementação deste trabalho (para traduzir o código). Não serão utilizados MACROS. A linguagem hipotética é formada por um conjunto de instruções e diretivas mostradas na Tabela no final desta especificação.

O programa deve entregar uma saída. Um arquivo em formato texto (arquivo.s) que deve ser a tradução do programa de entrada em Assembly IA-32. Observe que o certo seria 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 (section BSS, DATA ou TEXT). Porém para facilitar a segunda parte do

trabalho, a instrução SPACE deve ser traduzida na seção DATA sendo iniciada com valor zero. Este arquivo de sáida deve ser capaz de montar, ligar e rodar utilizando o sistema operacional LINUX, e o montador NASM.

Para entrada e saída de dados a linguagem hipotética, possui como sempre a leitura/escrita de números inteiros mediante as instruções INPUT, OUTPUT. Essas instruções devem ser capaz de ler números inteiros COM SINAL, com dígitos suficientes para ler valores de até 32 bits, e deve aceitar somente digitos entre 0 e 9 e o sinal de negativo no ínicio. Além disso, agora para ler/escrever caracteres existem as instruçõeses C_INPUT e C_OUPUT. As quais trabalham com um único caracter em ASCII. Para trabalhar com STRINGS as instruções S_INPUT e S_OUTPUT. As instruçõeses com string possuem 2 operados, além do endereço de memória onde deve ser lido/escrito o string, o outro operando é o tamanho do string. Finalmente, existem agora as instruções H_INPUT e H_OUPUT, as quais servem para trabalhar com números hexadecimal de até 32 bits. Essas instruções devem receber até 8 digitos de 0 a F, assumindo somente números positivos. Na saída do programa traduzido, deve existir então 8 sub-rotinas LeerInteiro, EscreverInteiro, LeerChar, EscreverChar, LeerString, EscreverString, LerHexa e EscreverHexa. As funçõoes devem estar em Assembly IA-32. No arquivo de saida (arquivo.s) as instruções de INPUT, OUTPUT, C_INPUT, C_OUPUT, S_INPUT, S_OUTPUT, H_INPUT e H_OUTPUT devem ser trocadas por chamadas a sub-rotinas mediante o comando CALL como visto em sala de aula. As funções LeerInteiro, EscreverInteiro, LeerChar, EscreverChar, LerHexa e EscreverHexa. Quando são lidos/escritos mais de um caracter/digito isso deve ser feito até o ENTER (0x0A) (no caso das funções de STRING elas devem esperar ou ENTER acontecer ou o tamanho máximo ser atingido). Todas as funções deve devolver em AX a quantidade de caracteres lidos/escritos. As funções NÃO podem ser cópias da io.mac (qualquer tentativa de cópia significa reprovação automática na disciplina como indicado no plano de ensino). As funções devem utilizar a pilha para receber argumentos.

DICA: Utilizar os registrados sempre como 32 bits (mesmo que a máquina hipotética de sala de aula era de 16 bits, já que neste trabalho as entradas de dados podem ter até 32 bits).

3.2 ELF-32

Realizar um programa ligador, chamado ligador_ia32.c, que receba o arquivo de saída da parte anterior (arquivo .s por linha de comando) e gere um arquivo executável em formato ELF-32. Note, que é necessário somente traduzir as 14 instruções equivalentes e os códigos para as funções de entrada e saída de dados (isso pode ser feito usando dissambly ou mediante ou manual da Intel). Para criação do ELF-32 é recomendado utilizar a bibleoteca ELFIO, também é permitido utilizar a LIBELF. A LIBELF é a blibleoteca mais utilizada para criação de arquivos ELF-32 porém é mais dificil de utilizar, a ELFIO é bem mais fácil de usar.

4 Avaliação

O prazo de entrega do trabalho é 3 de Julho de 2017. A entrega consistirá em:

- Código-fonte completo e comentado com instruções de compilação dos programas de tradução e simulação;

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]
C_INPUT	1	15	2	MEM[OP] <- STDIN
C_OUTPUT	1	16	2	STDOUT <- MEM[OP]
H_INPUT	1	17	2	MEM[OP] <- STDIN
H_OUTPUT	1	18	2	STDOUT <- MEM[OP]
S_INPUT	2	19	3	MEM[OP1] <- STDIN
S_OUTPUT	2	20	3	STDOUT <- MEM[OP1]
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