2º Trabalho Prático
CIC 116432 – Software Básico
Prof. Bruno Macchiavello
1º Semestre de 2020

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.c deve receber como entrada um arquivo .asm na linguagem hipotética de sala de aula, seguindo a especificação do Trabalho 1 com as seguintes diferenças:

- Space pode receber argumento
- Os rótulos agora podem utilizar o formar ROT + 1 (com espaço entre cada token)
- não precisa fazer macros
- não precisa fazer deteção de erro
- EQU e IF devem ser avaliados ANTES do programa ser traduzido a IA-32
- seráo colocadas mais intruções de I/O

O tradutor deve entregar como saída um arquivo em formato texto (arquivo.s) que deve ser a tradução do programa de entrada em Assembly IA-32. 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 (section BSS, DATA ou TEXT). Este arquivo de sáida deve ser capaz de montar, ligar e rodar utilizando o sistema operacional LINUX, e o montador NASM. Assumir sempre dados de e registradores de 32 bits. Lembrar de estender o sinal para o EDX antes de fazer operação de divisão e na multiplicação somente verificar se o resultado da multiplicação precisou mais de 32 bits e nesse caso dar mensagem de overflow. Lembrar de extender o sinal para o EDX antes de fazer a divisão. Assumir sempre tradução COM SINAL.

Para entrada e saída de dados devem ser implementadas funções em IA-32. 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 números de até no máximo 3 digitos. Além disso, agora para ler/escrever caracteres agora 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. Assumir que as strings nunca serão maiores que 100 caracteres (a instrução deve ler o string até chegar no tamanho indicado ou até o usuário digitar ENTER). Na saída do programa traduzido, deve existir então 8 sub-rotinas LeerInteiro, EscreverInteiro, LeerChar, EscreverChar, LeerString, EscreverString. 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 e S_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. 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 EAX a quantidade de caracteres lidos/escritos e mostrar na tela após a leitura do usuário a mesnagem: "Foram lídos YY caracteres". As funções NÃO podem ser cópias da io.mac (qualquer tentativa de cópia significa reprovação automática na disciplina como indicado no plano de ensino). É obrigatório o uso de PILHA para passagem de parâmetros.

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]
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