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

1 Introdução

O trabalho consiste em implementar em C/C++ um método de tradução de uma linguagem de montagem simples para uma representação de código objeto e IA-32.

2 Objetivo

Fixar o funcionamento de um processo de ligação e montagem. Na parte opcional os alunos podem aperfeiçoar o seu conhecimento sobre 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 hipotética vista em sala de aula (com algumas modificações, a serem descritas posteriormente). Sendo que deve estar separadas em seções de dados e códigos. Não será avaliado detecção de erros léxicos, semânticos ou sintáticos, porém é fortemente recomendado que os alunos utilizem como base o trabalho realizado anteriormente. A linguagem hipotética é formada por um conjunto de apenas 18 instruções (Tabela no final), e algumas diretivas.

O programa deve entregar três saídas. A primeira 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. A segunda saída 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 o código máquina (OPCODES e operandos) sem nenhum tipo de formatação. A terceira saída do programa *arquivo_debug.txt* deve ser os mesmo opcodes porém em formate texto e com espaço entre cada instrução (sem quebra de linhas).

Para entrada e saída de dados a linguagem hipotética, possui como sempre a leitura/escrita de números inteiros mediante as instruções INPUT, OUTPUT. Porém, agora para ler/escrever caracteres agora existem as instruções C_INPUT e C_OUPUT. E para trabalhar com STRINGS as instruções S_INPUT e S_OUTPUT. As instruções com caracteres funcionam exatamente igual as de inteiro com a diferença que o número este em ASCII, já as instruções com string possuem 1 operado, porém deve ler mais de um caractere. No seu programa, deve existir então 4 sub-rotinas *LeerInteiro*, *EscreverInteiro*, *LeerChar*, *EscreverChar*, *LeerString*, *EscreverString*. As funções 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* e *EscreverChar* devem ler ou escrever um único digito/letra. A função *LeerString* deve ler vários caracteres do teclado, até o ENTER (0x0A) ser digitado, deve devolver em AX a quantidade de caracteres lidos. Se durante a leitura faz a tentativa de ler um caractere a mais da quantidade de caracteres reservados em memória a função deve dar uma mensagem de erro: "Tentativa de acesso a memória não reservada", retornar 0 em AX e sair do programa. A função *EscreverString* deve escrever um caractere por caractere apontados pela memória até encontrar o final do string. As funções NÃO podem ser cópias da io.mac.

3.2 Parte Opcional

Valendo 1 ponto da media das provas. 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. 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 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 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. Lembre que as diretivas são avaliadas durante montagem/ligação e não geram código objeto.

Para as sub-rotinas de entrada de dados, o programa pode já ter as sub-rotinas pre-compiladas e somente copiar o código quando forem chamadas.

4 Avaliação

O prazo de entrega do trabalho é 1 de Dezembro de 2014. 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	12	2	MEM[OP] <- STDIN
C_OUTPUT	1	13	2	STDOUT <- MEM[OP]
S_INPUT	1	12	2	MEM[OP] <- STDIN
S_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
MACRO	0	-	0	Marcar início de suma MACRO. Sempre dentro da seção TEXT e antes do código principal
END	0	-	0	Marcar o fim de uma MACRO.