1º Trabalho Prático
CIC 116432 – Software Básico
Prof. Diego de Freitas Aranha
2º Semestre de 2013

1 Introdução

O trabalho consiste em implementar em C/C++ um método de tradução de uma linguagem de montagem simples para uma representação intermediária adequada para simulação em uma arquitetura moderna.

2 Método

A linguagem de montagem utilizada será a linguagem simbólica hipotética apresentada em sala, consistindo em um conjunto de apenas 14 instruções, com programas divididos em seções de código e dados. A linguagem hipotética não é sensível ao caso, não havendo diferenciação entre maiúsculas e minúsculas. Para cada instrução da máquina hipotética, a Tabela 1 abaixo contém o mnemônico, quantidade de operandos, código de operação utilizado na montagem, tamanho em palavras da instrução montada e uma breve descrição da sua utilidade. As linhas finais da tabela definem as diretivas para alocação de memória no segmento de dados.

Os identificadores de variáveis e rótulos são limitados em 1000 caracteres e seguem as regras comuns da linguagem C, sendo compostos por letras, números ou o caractere _ (*underscore*) e com a restrição de que o primeiro caractere não pode ser um número.

Para eliminar ambiguidade, as seções de código e dados devem ser devidamente marcadas com as diretivas correspondentes e aparecer no máximo uma vez mas em qualquer ordem, como ilustra o exemplo abaixo:

```
SECTION TEXT
ROT: INPUT N1
     COPY N1, N4 ; comentario qualquer
     COPY N2, N3[0]
     COPY N3[0], N3[1]
     OUTPUT N3[1]
     STOP
```

```
SECTION DATA
N1:  SPACE
N2:  CONST -0x10
N3:  SPACE 2
N4:  SPACE
```

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	Reservar memória não-inicializada para armazenamento de uma palavra.
SPACE	1	-	N	Reservar memória não-inicializada para armazenamento de um vetor do tamanho especificado.
CONST	1	-	1	Reservar memória para armazenamento de uma constante inteira de 16 *bits* em base decimal ou hexadecimal.

Tabela 1: Intruções e diretivas.

O programa de tradução deve ser capaz de realizar as fases de análise e síntese, mantendo informação intermediária armazenada em estruturas de dados e interrompendo a execução mediante a ocorrência de qualquer erro. A escolha apropriada de estruturas de dados faz parte do escopo do trabalho. O tratamento de erros deve abranger a lista não-exaustiva abaixo:

- Erros léxicos causados pela presença de caracteres inválidos no programa de entrada;
- Erros sintáticos causados pela presença de linhas que desviam por qualquer motivo da gramática elementar vista em sala;
- Erros semânticos causados por declarações ausentes ou repetidas, desvios condicionais e incondicionais para endereços na seção de dados, símbolos na seção de código como operandos de dados, escrita em memória reservada para armazenamento de constantes, divisão por constantes inicializadas com valor zero e indexação inválida de vetores. Observe que muitos montadores não restringem operandos de desvios ou escritas, mas adotaremos essa restrição para fins didáticos.

O programa de tradução deve receber dois argumentos em linha de comando, um arquivo de entrada contendo um programa em *Assembly* na linguagem hipotética e um arquivo de saída para armazenar o código objeto resultante. O formato do código objeto gerado pelo processo de tradução deve ser a simples concatenação de todos os inteiros de 16 *bits* que representam as instruções e seus operandos na seção de código, seguido pela seção de dados contendo todas as variáveis (vetores ou não) e constantes de 16 *bits* declaradas no programa de entrada. Por convenção e apesar da diretiva `SPACE` não garantir a inicialização de variáveis, utilizaremos o valor 0 para representar as posições de memória alocadas com essa diretiva.

O programa de simulação deve receber um único argumento em linha de comando, o arquivo objeto gerado pelo programa de tradução. A simulação deve ser capaz de conservar o funcionamento correto do programa de entrada e encerrar execução na primeira ocorrência de uma instrução `STOP`. Os únicos erros tratados pelo programa de simulação são a divisão por zero e o fornecimento de valores inválidos para a instrução `INPUT`.

3 Avaliação

O prazo de entrega do trabalho é 18 de Outubro de 2013. 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 por *e-mail*, de preferência utilizando o rótulo [SB]. Todas as entregas receberão mensagem de confirmação. O trabalho pode ser feito individualmente ou em dupla e será premiado o trabalho que obtiver o melhor desempenho

na combinação total do tempo de execução consumido pelas fases de tradução e simulação. A análise de desempenho será realizada pelo professor com programas de *benchmarking* suficientemente extensos, utilizando o mesmo nível de otimização `-O2` para a compilação de todos os trabalhos, e apenas se aplicará a trabalhos que produzem o resultado correto dos programas de *benchmarking*. Desta forma, a bonificação total do trabalho é de no máximo 2 pontos, divididos em até 2 alunos diferentes, a serem aplicados exclusivamente na primeira prova.

4 Dicas

Algumas funções recomendadas da biblioteca padrão da linguagem de programação C que podem reduzir substancialmente o trabalho envolvido são `fread()` e `fwrite()` para leitura e escrita do código intermediário e `strtok()` para implementação do analisador léxico. Os trabalhos serão executados em uma máquina equipada com Intel Core i5 540M com dois núcleos de processamento.