UNIVERSIDADE FEDERAL DE MINAS GERAIS

INSTITUTO DE CIÊNCIAS EXATAS

DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

IMPLEMENTAÇÃO DO LIGADOR DA MÁQUINA DE KHATTAB

Marlon Junior Barbosa Marques – 2012049839

BELO HORIZONTE
2013

INTRODUÇÃO

A *Máquina de Khattab* é uma máquina fictícia criada apenas para a disciplina Software Básico, do curso de Ciência da Computação da UFMG. Ela tem as seguintes características:

- ➔ A menor unidade endereçável nessa máquina é um inteiro;
- ➔ Os tipos de dados tratados pela máquina também são somente inteiros;
- ➔ A máquina possui uma memória de não menos que 1000 posições, 3 registradores de propósito específico e 8 registradores de propósito geral;
- ➔ Os registradores de propósito específico são:
  - – PC (contador de programas): contém o endereço da próxima instrução a ser executada;
  - – SP (apontador da pilha): aponta para o elemento no topo da pilha;
  - – PSW (palavra de status do processador): consiste em 2 bits que armazenam o estado da última operação lógico/aritmética realizada na máquina, sendo o bit mais significativo para indicar que a última operação resultou em zero, e outro bit para indicar que a última operação resultou num resultado negativo;
- ➔ Os registradores de propósito geral são indexados por um valor que varia de 0 a 7;
- ➔ A única forma de endereçamento existente na máquina é direto, relativo ao PC;
- ➔ As instruções READ e WRITE lêem e escrevem um inteiro na saída padrão do emulador;
- ➔ As instruções são codificadas em um inteiro, podendo ter dois, um ou nenhum operando, que é o caso das instruções RET e HALT.
- ➔ Os operandos podem ser uma posição de memória (M, codificado como inteiro) ou um registrador (R, codificado como um inteiro entre 0 e 7)

O conjunto de instruções da Máquina de Khattab está detalhado na Tabela 1. As operações marcadas com * atualizam o PSW.

Cód	Símbolo	Operandos	Significado	Ação
01	LOAD	R M	Carrega Registrador	$Reg[R] \leftarrow Mem[M + PC]$
02	STORE	R M	Armazena Registrador	$Mem[M + PC] \leftarrow Reg[R]$
03	READ	R	Lê valor para registrador	$Reg[R] \leftarrow$ "valor lido"
04	WRITE	R	Escreve conteúdo do registrador	"Imprime" $Reg[R]$
05	COPY	R1 R2	Copia registrador	$Reg[R1] \leftarrow Reg[R2]$ *
06	ADD	R1 R2	Soma dois registradores	$Reg[R1] \leftarrow Reg[R1] + Reg[R2]$ *
07	SUB	R1 R2	Subtrai dois registradores	$Reg[R1] \leftarrow Reg[R1] - Reg[R2]$ *
08	AND	R1 R2	AND(bit a bit) de dois registradores	$Reg[R1] \leftarrow Reg[R1]$ AND $Reg[R2]$ *
09	OR	R1 R2	OR(bit a bit) de dois registradores	$Reg[R1] \leftarrow Reg[R1]$ OR $Reg[R2]$ *
10	XOR	R1 R2	XOR(bit a bit) de dois registradores	$Reg[R1] \leftarrow Reg[R1]$ XOR $Reg[R2]$ *
11	NOT	R1	NOT(bit a bit) de um registrador	$Reg[R1] \leftarrow$ NOT $Reg[R1]$ *
12	JMP	M	Desvio incondicional	$PC \leftarrow PC + M$
13	JZ	M	Desvia se zero	Se $PSW[zero]$, $PC \leftarrow PC + M$
14	JNZ	M	Desvia se não zero	Se $!PSW[zero]$, $PC \leftarrow PC + M$
15	JN	M	Desvia se negativo	Se $PSW[negativo]$, $PC \leftarrow PC + M$
16	JP	M	Desvia se positivo	Se $!PSW[negativo]$ e $!PSW[zero]$, $PC \leftarrow PC + M$
17	PUSH	R	Empilha valor do registrador	$SP \leftarrow SP - 1$ $Mem[SP] \leftarrow Reg[R]$
18	POP	R	Desempilha valor no registrador	$Reg[R] \leftarrow Mem[SP]$ $SP \leftarrow SP + 1$
19	CALL	M	Chamada de subrotina	$SP \leftarrow SP - 1$ $Mem[SP] \leftarrow PC$ $PC \leftarrow PC + M$
20	RET		Retorno de subrotina	$PC \leftarrow Mem[SP]$ $SP \leftarrow SP + 1$
21	HALT		Parada	

Tabela 1: Instruções da Máquina de Khattab

Além das instruções descritas na Tabela 1, também estarão presentes duas pseudoinstruções:

– WORD A → Usada para alocar uma posição de memória cujo valor será A, ou seja, quando houver tal instrução, a posição de memória que está sendo referenciado pelo PC deverá receber o valor A, que é um inteiro.

– END → Indica o final do programa para o montador.

Além do mais, a linguagém simbólica do assembler possui o seguinte formato:

[<label>:] <operador> <operando1> <operando2> [; comentário]

Espaços em branco não interferem com a tradução do programa.

O padrão para definição de macros é demonstrado no seguinte exemplo:

PRINT: BEGINMACRO

WRITE R0

WRITE R1

WRITE R2

ENDMACRO

O exemplo acima, de simples entendimento, mostra que sempre que a palavra-chave BEGINMACRO aparecer, significa que todas as linhas até que a outra palavra-chave ENDMACRO apareça, correspondem ao corpo da macro. O nome da macro vem antes de BEGINMACRO, seguido de ":". O processo de expansão corresponde ao fato de substituir todas as expressões onde aparecem o nome da macro pelo seu corpo, nas linhas do código.

Um detalhe importante é que a macro pode ou não apresentar parâmetro (no máximo 1). No caso de apresentar parâmetro, ele deve ser substituído pelo argumento passado na hora de expandir as macros.

O objetivo deste trabalho será a implementação de um editor de ligação (linker) da Máquina de Khattab escrito na linguagem C++ de programação. O linker será implementado através da estratégia de três passos, alocação, ligação e relocação, descritas por Tanenbaum[1]. A entrada do linker serão os arquivos objetos que resultaram do processo de assembler, e a saída será o arquivo executável da Máquina de Khattab (a ser executado no TP1).

Uma observação importante que deve ser notada é que para garantir o funcionamento deste trabalho, o TP2 (montador) teve de ser ligeiramente modificado. Agora, ao invés de dar como saída um executável já pronto para o emulador (arquivo .mk), ele fornecerá a saída para este trabalho, que será um arquivo objeto em forma de texto (arquivo .mmk). Este arquivo conterá a tabela de símbolos do módulo, a tabela de referências (que informa onde no arquivo ocorre a referência a um símbolo, ou se esta referência não se encontra lá, informa um código especial), e finalmente, as próprias instruções do módulo, decodificadas como inteiros (linguagem de máquina). Tal mudança foi realizada facilmente, e o código adaptado está disponível junto com esta documentação.

ABSTRAÇÃO E IMPLEMENTAÇÃO

A fim de se realizar a ligação dos objetos gerados pelo montador(.mmk), foi criada uma classe *linker*, cuja único método público é seu construtor, que recebe como parâmetros várias strings contendo o nome de cada arquivo módulo, e o arquivo de saída. Os principais métodos e variáveis privadas utilizadas por esta classe para realizar a expansão são os seguintes:

- struct object

```
{
   int memory[SIZE], begin, end, size;
   multimap<string, int> symTable, refTable;
};
```

Como dito acima, cada objeto conterá uma tabela de símbolos e de referências, que foram representadas no programa pelo container da STL *multimap*. As variáveis begin, end e size, marcam o início, o fim, e o tamanho do módulo, enquanto que o vetor memory corresponde

- object *modules;

  Vetor de objetos que representam os módulos passados como argumento do programa

- int numObj;

  Número de objetos passados como argumento do programa

- void ReadObject(string fileName, object &o)

  Dado uma string contendo o nome do objeto, lê todos os seus dados (tabela de símbolos, tabela de referência e dados) para o vetor de objetos da classe linker.

- void linker:: AllocateObjects()

  Faz a alocação do editor de ligação. Atribui o valor de begin e end para cada objeto

- void linker:: ReAllocateObjects()

  Sai vasculhando a tabela de símbolos e de referências de cada objeto e faz a relocação sempre que encontrar referências externas.

Todas as funções necessárias para a execução do programa estão definidas e declaradas nos arquivos linker.*h* e linker.*cpp*, os quais foram muito bem comentados e são de fácil entendimento.

COMPILAÇÃO E EXECUÇÃO

Com o intuito de facilitar a compilação, foi incluído um arquivo *Makefile* junto ao código. Para compilar, basta digitar o comando *make* no diretório do código, em um ambiente UNIX like ou qualquer outro que contenha o programa *GNU Make*. O *make* gerará o arquivo binário *linker*.

A pasta do programa é dividida em vários diretórios: *bin* (onde fica o executável), *build* (onde ficam os arquivos de objeto), *doc* (onde fica a documentação), *src (*onde ficam os arquivos-fonte) e *tst* (onde ficam os arquivos de teste).

Para executar o linker propriamente, o executável espera que sejam passados no mínimo 4 argumentos, além do próprio editor de ligação. Na realidade, são esperados n+2 argumentos, onde n é o número de módulos a serem ligados. Por exemplo:

./bin/linker modulo1.mmk modulo2.mmk modulo3.mmk -m main.mmk -o programa.mk

A flag -m indica que o próximo argumento será o arquivo principal do programa e a flag -o significa que o próximo argumento é a saída.

É importante ressaltar que tanto a entrada do programa são arquivos objeto (em texto), que representam os módulos já traduzidos, e a saída é um arquivo binário.

Outros possíveis comandos do *Makefile* são:

- *make build*: compila os arquivos fonte, gerando arquivos objeto e o binário (a mesma coisa que *make* sozinho faria).

-*make clean:* exclui os arquivos objeto, temporários, e o executável.

Para fins de completude, foram incluídos os arquivos-fonte de todos os TPs até então (com o TP2 modificado). O arquivo make compilará todos os TPs, e o usuário pode executar aquele que desejar.

TESTES

Na especificação do trabalho pediu-se para implementar dois testes para o ligador, e os dois foram implementados com sucesso. São eles: uma **calculadora** e um determinador de **números primos**. Eles funcionam da seguinte forma:

Calculadora: Programa que lê três números (a, b, c) e realiza as operações adição, subtração, multiplicação, divisão inteira (imprimindo o quociente e o resto) e exponenciação. Onde:

– b é a operação a ser realizada: b=1: adição; b=2: subtração; b=3: multiplica-

ção; b=4: divisão inteira (imprimindo o quociente e o resto); b=5:exponenciação;

- a e c são os valores aos quais a operação será realizada;
- Arquivos envolvidos: calcMain.amk, addModule.amk, subModule.amk, mulModule.amk, divModule.amk e expModule.amk

Números primos: Progama que lê um número da entrada padrão e imprime o primeiro número primo maior que este na saída padrão.

- Arquivos envolvidos: divModule.amk e primeMain.amk

Todos os arquivos de entrada de teste, (.amk) e suas respectivas saídas (.mmk e .mk), estão no diretório tst/ deste trabalho. Todos os arquivos acima foram testados montador adaptado do TP2 e no emulador criado no TP1 e funcionaram perfeitamente. A figura 1 demonstra o processo de maneira clara.

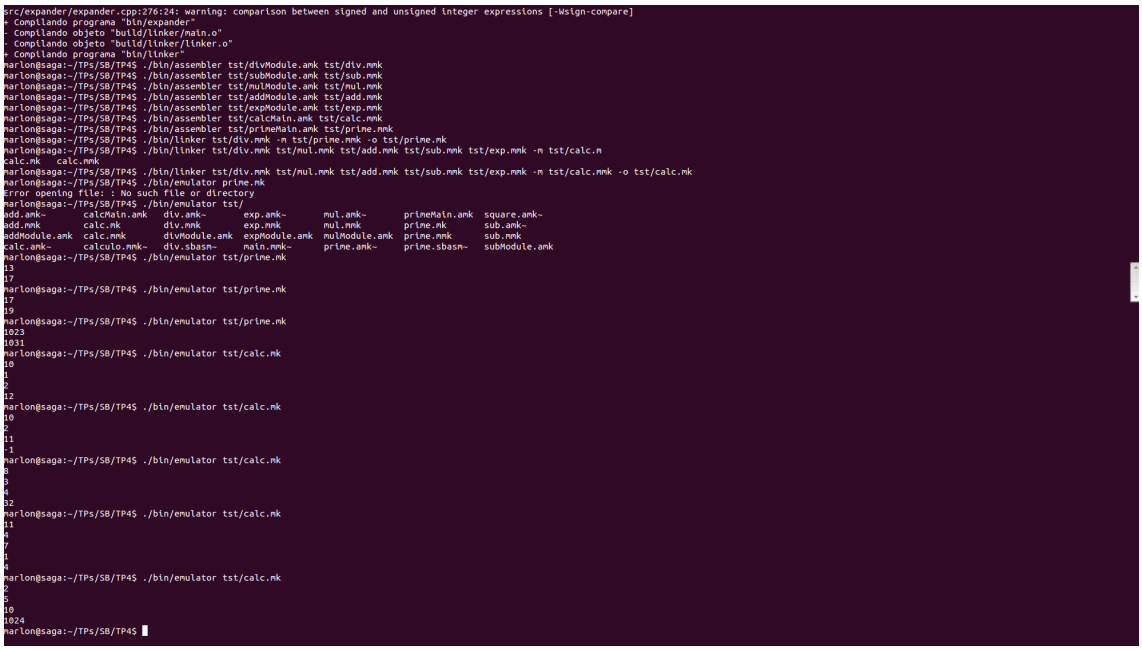

Figura 1: Entrada: vários módulos em linguagem de montagem . Saída: arquivo executável da máquina de Khattab (da calculadora e dos números primos)

CONCLUSÃO

O editor de ligação da Máquina de Khattab, abstração para fixar os conceitos da disciplina Software Básico, foi implementado com sucesso. Este foi o trabalho final da matéria, e com ele foi possível entender o funcionamento de uma máquina simplificada, passando pelo mais alto nível (a linguagem de montagem), até o mais baixo nível (linguagem de máquina). A Figura 2 demonstra esse processo. Finalmente, todos os trabalhos foram testados, e funcionaram corretamente: conseguiu-se a façanha de literalmente “construir um programa”, conhecendo passo a passo os processos envolvidos.

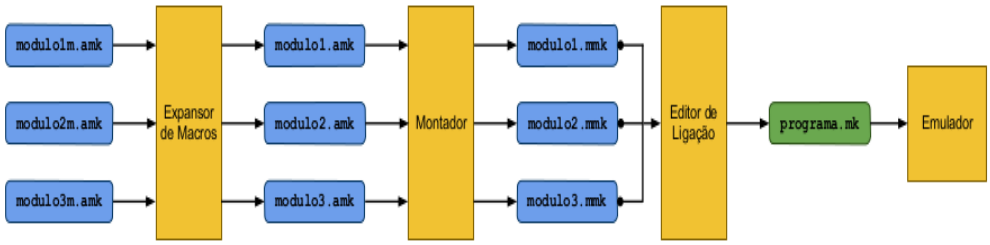

Figura 2: Esquema que representa os níveis da máquina de Khattab, desde o código em assembler até a execução da linguagem de máquina

BIBLIOGRAFIA

1: Tanenbaum, Andrew S. *Structured Computer Organization*. Prentice Hall; 5 edition (June 25, 2005).