CMP1057 - Arquitetura de Computadores I
Max Gontijo de Oliveira
1º Trabalho
Emulador de Máquina de Von Neumann

1 Resumo

Nesse trabalho, deverá ser construído um emulador básico e simples de uma máquina baseada na arquitetura de Von Neumann.

2 Detalhes

Deverá ser criado um programa em qualquer linguagem de programação que seja capaz de emular o funcionamento de uma máquina conforme será descrita a seguir.

2.1 Definição da máquina

A máquina que deverá ser emulada é baseada em uma versão simplificada do modelo proposto por Von Neumann. Assim, essa máquina terá uma unidade lógica e aritmética (ULA), uma unidade de controle (UC), uma memória principal. O modelo de Von Neumann prevê suporte à unidades de entrada e saída. Contudo, nesse trabalho, esse suporte não precisará ser implementado.

A máquina que deseja-se emular tem uma memória principal com 1000 linhas (palavras). Cada palavra é composta por até 6 dígitos decimais.

O emulador desenvolvido deverá ser capaz de carregar um arquivo de entrada em toda a sua memória. Esse arquivo deverá representar fielmente cada palavra da memória, da primeira à última. Após carregar o arquivo de entrada, o emulador deverá iniciar o processo de execução executando a primeira instrução, considerando que ela estará presente na primeira palavra da memória principal. Após a execução da instrução, o emulador deverá executar a instrução seguinte e assim por diante. Ou seja, se terminou de executar a instrução localizada na posição 45 da memória principal, a próxima instrução a ser buscada e executada estará localizada na posição 46. Exceto nos casos em que a instrução for um desvio, caso em que a próxima instrução a ser executada deverá ser aquele para onde o desvio ocorre. Os detalhes do fluxo de execução podem ser vistos na Seção 2.1.1; na Seção 2.1.2 será apresentada a estrutura da memória principal; o conjunto de instruções que o emulador deverá prover será descrito na Seção 2.1.3.

Antes de continuar, porém, é necessário conhecer quais são os registradores que estarão presentes na ULA e na UC.

Na ULA haverão três registradores: AC, MQ e MBR. Os registradores AC e MQ serão somente acessíveis de dentro da própria ULA. Eles servem como parâmetros de entrada e saída para as operações lógicas e aritméticas. Por exemplo, a instrução de adição pode ser resumida em $AC \leftarrow AC + MQ$. O registrador MBR está localizado na ULA, mas está conectado também com a UC e com a memória principal. Esse registrador é o ÚNICO meio de se receber alguma palavra da memória principal ou escrever alguma palavra na memória principal. Dessa forma, em qualquer instrução que demandar a leitura ou a escrita na memória, a palavra a ser escrita/lida deverá, antes, passar por MBR.

Na UC, haverão três registradores: PC, IR e MAR. O registrador PC é um registrador de endereçamento. Ele contém o endereço da próxima instrução que será buscada e executada. Inicialmente ele é zerado, pois a primeira instrução a ser buscada e executada deverá estar na primeira linha da memória principal. O registrador IR é o registrador de instrução. Ele é o responsável por armazenar o código da instrução que será executada. O registrador MAR é o registrador de endereçamento direto à memória principal. Junto com o registrador MAR, são os ÚNICOS registradores com acesso direto à memória principal.

Todo acesso à memória principal deve ser realizado por meio dos registradores MAR e MBR. Em uma operação de leitura, MAR precisa receber o endereço que será lido na memória, enquanto MBR irá receber o valor que está armazenado nesse endereço apontado por MAR. Em uma operação de escrita, o registrador MBR deve receber o valor a ser armazenado enquanto MAR deve receber o endereço da palavra na memória principal onde tal valor será armazenado; após preencher corretamente os dois registradores, a memória pode ser atualizada.

2.1.1 Fluxo de execução

O fluxo de execução principal esperado no emulador a ser desenvolvido pode ser visto no Algoritmo 1. Nesse algoritmo, considere que $M1(PC)$ se refere aos primeiros 3 dígitos (decimais) do valor armazenado na posição PC da memória principal, enquanto $M2(PC)$ se refere aos últimos 3 dígitos (decimais) do valor armazenado na posição PC da memória principal. Por exemplo, se $PC = 37$ e a palavra 37 da memória principal tiver o valor 937402, então $M1(PC) = 937$ e $M2(PC) = 402$.

Dica de programação: para calcular $M1(PC)$ e $M2(PC)$ é muito fácil. Dado que $M(PC)$ seja o valor da palavra toda localizada na posição PC da memória principal, $M1(PC) = \lfloor M(PC) \div 1000 \rfloor$ (quociente da divisão) e $M2(PC) = M(PC) \bmod 1000$ (resto da divisão).

Algoritmo 1: Emulador de máquina de Von Neumann

Entrada: arquivo contendo um conjunto M de 1000 linhas

```
1  PC ← 0
2  enquanto PC ≠ −1 faça
3  |  IR ← M1(PC)
4  |  Identifica qual é a instrução a ser executada
5  |  se IR é um código de uma instrução que tem operando então
6  |  |  MAR ← M2(PC)
7  |  fim
8  |  PC ← PC + 1
9  |  Executa a instrução cujo código está em IR
10 fim
11 Emulador escreve em um arquivo de saída o conteúdo da memória principal
12 Emulador finaliza execução
```

A estrutura de repetição entre as linhas 2 e 10 se referem ao ciclo de execução de uma instrução, compreendendo a busca da instrução (linha 1), a busca de operandos (linha 6) e a efetiva execução da instrução (linha 9). Na linha 1, o registrador PC é iniciado com 0, pois a primeira instrução a ser executada pelo emulador estará nessa posição da memória principal. Na linha 3 o emulador deve carregar em IR o código da instrução da linha PC da memória principal. Na linha 6, o emulador deve carregar em MAR parâmetro da instrução da linha PC da memória principal. Note que o carregamento é em MAR porque todas as instruções com operandos deverão buscar seus operandos na memória principal. Na linha 8 o registrador PC é incrementado, pois, normalmente, a próxima instrução a ser executada estará na linha seguinte; contudo, se a instrução executada na linha 9 for um desvio, esse valor de PC será modificado. A linha 9 se refere à execução da instrução; o conjunto de instruções que o emulador deve prover será descrito na Seção 2.1.3. A linha 11 não se refere à uma operação de E/S da máquina emulada; trata-se apenas do resultado final que será avaliado pelo professor.

Embora nesse algoritmo as linhas que se referem ao carregamento de algum valor localizado na memória principal não detalhem o que de fato deve ocorrer, lembre-se de que QUALQUER acesso de leitura à memória precisa antes passar pelo registrador MBR.

2.1.2 Memória principal

A máquina que deseja-se emular tem uma memória principal com 1000 linhas (palavras). Cada palavra é composta por até 6 dígitos decimais, sem contar com o dígito de sinal. Cada palavra da memória principal pode ser uma instrução ou uma palavra de dados. Se for uma instrução, a palavra terá SEMPRE seis dígitos, de modo que os três primeiros dígitos se referem ao código da instrução e os três dígitos restantes, ao parâmetro. Se a palavra for uma palavra de dados, todos os dígitos irão compor o valor. Por exemplo, seja a palavra 215074, se for uma instrução, 215 será o código da instrução e 074 (ou simplesmente 74) será o parâmetro; se for uma palavra de dados, o valor 215074 será o valor como um todo.

O arquivo de entrada será justamente a configuração inicial dessa memória principal. Dessa forma, a entrada será um arquivo de exatamente 1000 linhas, sendo que em cada linha haverá um número de seis dígitos (base decimal).

As linhas do arquivo que se referirem a instruções serão compostas sempre de um número de 6 dígitos decimais e mais nada. As linhas do arquivo que se referirem a dados poderão ter apenas seis dígitos decimais e mais nada (tal como as linhas que se referem a instruções), ou os seis dígitos e o sinal de menos (quando se tratar de um número negativo).

Segue um exemplo de um trecho aleatório de 10 linhas de um possível arquivo de entrada:

```
100749
310456
200315
543623
−000342
002500
304208
−793060
105689
110544
```

2.1.3 Instruções

A máquina que deseja-se emular possui um conjunto de instruções bastante simplificado. Possui instruções de transferência de dados, como leituras e escritas na memória e registradores; instruções de desvio condicional e incondicional; e instruções aritméticas.

O conjunto de instruções que o emulador deverá prover é descrito na Tabela 1. A maioria das instruções possuem um argumento. Esse argumento sempre será um endereço na memória. Note que as instruções estão descritas de forma

simplificada. Assim, não se esqueça das restrições de acesso à memória: leituras e escritas devem ser realizadas com o uso dos registradores MBR e MAR.

Código instrução	**Representação simbólica**	**Descrição**
100	CARREGAR_MQ(X)	Transfere o valor armazenado na memória na posição X para o registrador MQ.
105	CARREGAR_AC(X)	Transfere o valor armazenado na memória na posição X para o registrador AC.
110	ESCREVER_MQ(X)	Transfere o valor armazenado no registrador MQ para a memória principal no endereço X.
115	ESCREVER_AC(X)	Transfere o valor armazenado no registrador AC para a memória principal no endereço X.
120	COPIAR_MQ_AC( )	Transfere o valor armazenado no registrador MQ para o registrador AC.
125	COPIAR_AC_MQ( )	Transfere o valor armazenado no registrador AC para o registrador MQ.
200	DESVIAR(X)	Faz com que o X seja o endereço da próxima instrução a ser buscada, incondicionalmente.
205	DESVIAR-(X)	Faz com que o X seja o endereço da próxima instrução a ser buscada, caso $AC < 0$.
210	DESVIAR-=(X)	Faz com que o X seja o endereço da próxima instrução a ser buscada, caso $AC \leq 0$.
215	DESVIAR=(X)	Faz com que o X seja o endereço da próxima instrução a ser buscada, caso $AC = 0$.
220	DESVIAR+(X)	Faz com que o X seja o endereço da próxima instrução a ser buscada, caso $AC > 0$.
225	DESVIAR+=(X)	Faz com que o X seja o endereço da próxima instrução a ser buscada, caso $AC \geq 0$.
300	ADD(X)	Soma o valor armazenado em AC com o valor armazenado na posição X da memória principal e guarda o resultado em AC. Ou seja, $AC \leftarrow AC + M(X)$.
305	SUB(X)	Subtrai do valor armazenado em AC o valor armazenado na posição X da memória principal e guarda o resultado em AC. Ou seja, $AC \leftarrow AC - M(X)$.
310	MUL(X)	Multiplica o valor armazenado em MQ pelo valor armazenado na posição X da memória principal e guarda o resultado em AC. Ou seja, $AC \leftarrow MQ \times M(X)$.
315	DIV(X)	Divide o valor armazenado em AC pelo valor armazenado na posição X da memória principal; o quociente é armazenado em MQ e o resto em AC. Ou seja, $MQ \leftarrow \lfloor AC \div M(X) \rfloor$ e $AC \leftarrow AC \bmod M(X)$.
400	PARAR( )	Faz com que a máquina pare de buscar e executar instruções. Essa é a instrução que força a saída da estrutura de repetição (linhas 2 à 10) no Algoritmo 1.
500	CARREGAR_AC_ENDERECO_MQ( )	Transfere o valor armazenado na memória na posição MQ para o registrador AC. Ou seja, $AC \leftarrow M(MQ)$.
505	ESCREVER_AC_ENDERECO_MQ( )	Transfere o valor armazenado no registrador AC para a memória principal no endereço MQ. Ou seja, $M(MQ) \leftarrow AC$.

Tabela 1: Conjunto de instruções da máquina emulada

2.2 Grupos

O trabalho poderá ser feito individualmente ou em grupos de até três alunos.

3 Entrega

A entrega do trabalho consistirá no envio dos arquivos de código fonte e do arquivo de saída via e-mail para o professor com o seguinte assunto:

`[CMP1057] Trabalho AED - Máquina de Von Neumann`

O envio deverá ser realizado por apenas um membro do grupo e, no corpo do e-mail, **deve constar o nome completo de cada um dos membros**.

Observação: O servidor de e-mails do professor pode recusar arquivos considerados de risco. Isso pode acontecer com arquivos executáveis, de bibliotecas e outros que podem constar na lista negra. Isso vale para arquivos listados como de risco que estejam presentes dentro de um arquivo compactado. Portanto, quando enviar, caso tenha dificuldades, realize a compactação do arquivo utilizando .7Z ou .RAR com senha de encriptação "**123456**", encriptando, ainda, a lista de arquivos do arquivo compactado. Mas facilite a vida do seu professor. Só faça isso se você tiver algum problema para enviar seu código fonte. Tente antes compactar sem senha. Mas, repetindo: se for compactar com senha, utilize a senha "**123456**".

3.1 Código fonte

O programa pode ser escrito em qualquer linguagem de programação. Além dos códigos fontes, deve ser enviado um arquivo de texto contendo informações necessárias para compilação e execução do programa. Caso o programa seja escrito em uma linguagem que não possa ser compilada em ambiente Linux, é necessário enviar o executável também.

3.2 Arquivo de saída

O arquivo de saída será um arquivo de texto cujo conteúdo obedeça o mesmo formato do arquivo de entrada. De fato, trata-se das 1000 linhas da memória principal após a execução.

4 Avaliação

A avaliação será realizada pela análise do código fonte e do resultado obtido na execução.

Todos os membros do grupo receberão a mesma nota.

O professor se reserva no direito de estender a avaliação para questionamentos orais e pessoais junto aos alunos com trabalhos com suspeita de plágio.

Trabalhos com plágio identificado serão zerados.