Trabalho 2 (2024/2)

Disciplina de Programação Funcional

A) Data de entrega: 08/11/2024

B) Objetivo:

O objetivo deste trabalho é consolidar o conhecimento sobre construção de funções recursivas e programação interativa em Haskell.

C) Enunciado:

O objetivo desta tarefa é construir um programa Haskell capaz de simular um computador hipotético de arquitetura simplificada.
O programa deve receber como entrada uma lista, que representa a memória do computador simulado, previamente carregada com um programa a ser executado e os dados usados pelo mesmo. Como bônus (+1,0) de avaliação do trabalho, a leitura pode ser realizada a partir de um arquivo texto.
A simulação deve consistir na "execução" do programa carregado na memória e o resultado deve ser uma lista com o estado da memória após a simulação. A arquitetura do computador hipotético pode ser vista na figura 1.

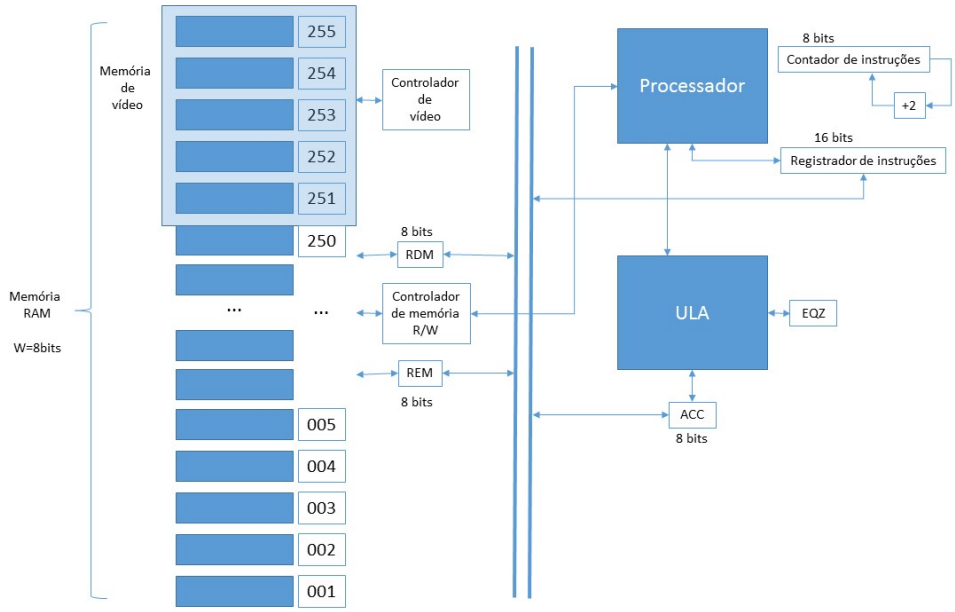

Figura 1 – Arquitetura do computador simulado

A memória possui 256 bytes endereçados de 0 a 255. Tanto o registrador de dados da memória (RDM) como o registrador de endereços da memória (REM) têm 8 bits. Os programas a serem executados devem ser carregados sempre a partir do endereço zero (endereço com o qual o contador de instruções é inicializado). As posições de memória de 251 a 255 correspondem a endereços da memória de vídeo. Qualquer valor inteiro de 8 bits armazenado nestes endereços terá seu valor correspondente em decimal exibido na "tela" do computador hipotético.

A Unidade Lógica e Aritmética possui apenas um registrador acumulador (ACC) de oito bits e um flag (EQZ) que indica quando o valor do acumulador é igual a zero. A ULA é capaz de somar ou subtrair o conteúdo de uma posição de memória com o conteúdo do acumulador, armazenando o resultado no próprio acumulador.

Por fim o processador possui um contador de instruções de 8 bits e um registrador de instruções de 16 bits. O registrador de instruções possui 16 bits porque todas as instruções (a exceção da instrução de parada) são compostas pelo código da instrução (8 bits) seguido de um endereço de memória (8 bits), de maneira que a carga de uma instrução corresponde a carga de 16 bits de cada vez (por esta razão o contador de instruções é incrementado de 2 unidades por vez). O conjunto de instruções pode ser visto na tabela 1.

Código (decimal)	**Instrução**	**Detalhamento**
02	LOD <end>	Carrega o conteúdo do endereço de memória <end> no registrador acumulador (ACC).
04	STO <end>	Armazena o conteúdo do registrador acumulador (ACC) no endereço de memória <end>.
06	JMP <end>	Desvio incondicional: carrega no contador de instruções o valor <end> forçando com que a próxima instrução a ser executada seja a que se encontra no endereço de memória <end>.
08	JMZ <end>	Desvio condicional: funcionamento análogo ao da instrução JMP com a diferença que a carga do contador de instruções só ocorre se o valor do acumulador for igual a zero (de acordo com a flag EQZ).
10	CPE <end>	Se o conteúdo do endereço <end> for igual ao acumulador, coloca 0 no acumulador, caso contrário coloca 1.
14	ADD <end>	Adiciona o conteúdo do endereço de memória <end> ao conteúdo armazenado no acumulador (ACC) e armazena a resposta no próprio acumulador.
16	SUB <end>	Subtrai o conteúdo do endereço de memória <end> do conteúdo do acumulador (ACC) e armazena a resposta no próprio acumulador.
18	NOP	Não executa ação nenhuma (*No OPeration*).
20	HLT	Encerra o ciclo de execução do processador (*HaLT*).

Tabela 1 – Conjunto de instruções do computador hipotético

Simplificações:

- A memória poderá ser representada em Haskell por uma lista de tuplas. Cada tupla é formada por dois inteiros, onde o primeiro representa o endereço de memória e o segundo o conteúdo daquela posição de memória. Desta forma pode-se inserir na lista informada por parâmetro apenas os endereços que serão trabalhados durante a simulação.
- Tanto os códigos de instrução como o conteúdo dos endereços de memória poderão ser informados em decimal, desde que respeitados o tamanho máximo de bits correspondentes ao valor binário associado.
- As operações aritméticas e lógicas podem ser feitas em decimal, desde que respeitados o tamanho máximo de bits correspondentes ao valor binário associado. Valores devem ser truncados de acordo.
- A ULA trabalha apenas com valores de 8 bits e não possui indicador de “carry”, de “overflow” ou de “underflow”.

Dicas:

- Devem ser modeladas estruturas de dados adequadas para representar a memória do computador, bem como o estado atual de cada registrador envolvido na computação.
- Convém modelar a execução de cada passo de computação básica e realizar a simulação como uma composição de cada passo básico.

Programas teste:
Para demonstrar o funcionamento do simulador os programas de 1 a 3, escritos em uma linguagem imperativa, deverão ser traduzidos para o assembler do computador hipotético e executados no mesmo. Nestes programas considere que a variável "Resp" corresponde ao endereço de memória 251, as variáveis A, B, C, D e E aos endereços 240, 241 e assim por diante até 244 e que valores constantes são armazenados nos endereços a partir de 245 na ordem em que aparecem.

1) Resp = A + B – 2;
2) Resp = A * B;
3) A = 0; Resp = 1; while(A < 5) { A = A + 1; Resp = Resp + 2; }

Como exemplo, o assembler para o programa "Resp = A + B" é tal como:
0 LOD 240
2 ADD 241
4 STO 251
6 HLT NOP

O documento de entrega da tarefa deverá conter o código Haskell do simulador, a definição das listas a serem usadas como entrada para cada um dos programas de teste definidos, bem como o assembler de cada um dos programas teste.

D) Desenvolvimento e avaliação do trabalho:

- O trabalho pode ser realizado individualmente ou em grupos de, no máximo, 3 alunos.
- Programas que não consigam ser executados receberão nota zero.
- Mensagens de erro apresentadas durante a execução do programa serão consideradas como erros de execução, e acarretarão descontos na nota do trabalho.
- Os trabalhos serão avaliados de acordo com critérios a serem estabelecidos pelo professor da disciplina, considerando o que é pedido no enunciado e o que foi realizado com sucesso.
- ***Trabalhos copiados resultarão em nota zero para todos os alunos envolvidos.***

E) Entrega do trabalho:

- Todos os arquivos-fonte e os arquivos de exemplo deverão ser empacotados em um único arquivo (.zip) e submetidos através do sistema Moodle até a data de entrega.
- Não serão aceitos trabalhos enviados por correio eletrônico.
- Não serão aceitos trabalhos enviados fora do prazo estabelecido.