Tarefa : Simulação de um computador hipotético

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. 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.

A memória possui 256 bytes endereçados de 0 a 255. Tanto o registrador de dados da memória como o registrador de endereços da memória 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.

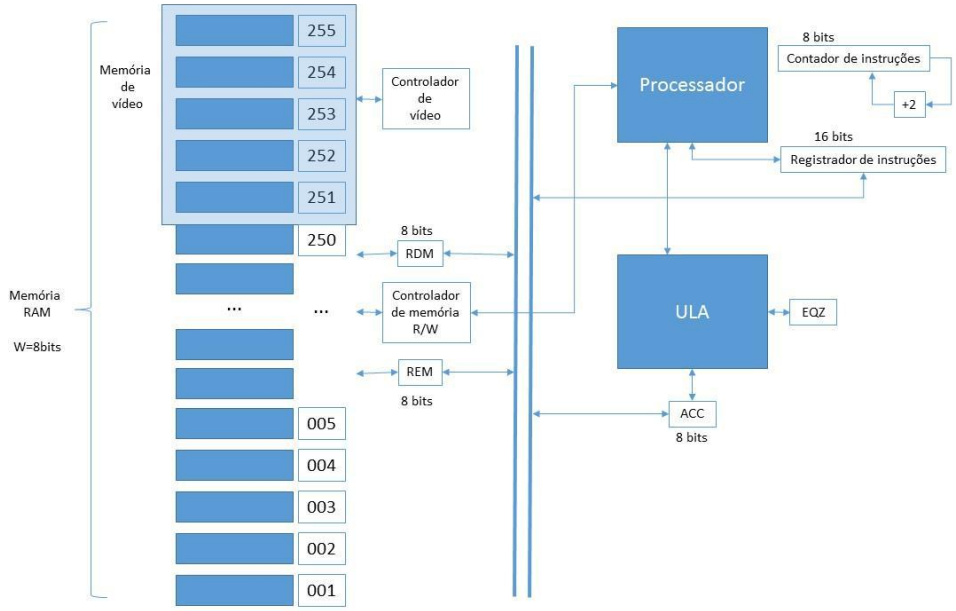

Figura 1 – Arquitetura do computador simulado

A Unidade Lógica e Aritmética possui apenas um registrador acumulador (ACC) e um flag que indica quando o valor de uma posição de memória é igual a zero. 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 seguido de um endereço de memória, 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	**Instrução**	**Detalhamento**
02	LOD <end>	Carrega o conteúdo do endereço de memória <end> no registrador acumulador.
04	STO <end>	Copia o conteúdo do registrador acumulador 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.
10	CPL <end>	Se o conteúdo do endereço <end> for menor que zero, coloca 1 no acumulador, em qualquer outro caso coloca 0.
12	AND <end>	Se o conteúdo do acumulador e da posição de memória <end> são ambos 0, coloca 1 no acumulador, em qualquer outro caso coloca 0.
14	ADD <end>	Adiciona o conteúdo do endereço de memória <end> ao conteúdo armazenado no acumulador e armazena a resposta no próprio acumulador.
16	SUB <end>	Subtrai o conteúdo do endereço de memória <end> do conteúdo do acumulador 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 por uma lista de tuplas. Cada tupla é formada por dois inteiros de 8 bits, onde o primeiro representa o endereço de memória (sem sinal) e o segundo o conteúdo daquela posição de memória (com sinal). 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.

- As operações aritméticas e lógicas podem ser feitas em decimal.
- A ULA trabalha apenas com valores de 1 byte e não possui indicador de “carry”. Valores maiores que 255 simplesmente são armazenados como zero.

Observação: na implementação do simulador todos os laços deverão ser implementados usando recursividade.

Programas teste

Para demonstrar o funcionamento do simulador os programas de 1 a 4, escritos em Java, 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) If (A > B) resp = A – B else resp = B – A
4) A = 0; Resp = 1; While(A < 10) { A = A + 1; Resp = Resp + 2; }

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.