Memória (1)

Contém uma série de posições, cada uma contendo um valor. Cada posição é identificada por um número, iniciando por 0, chamado endereço. No início, várias dessas posições já contêm valores, que representam o programa a ser executado. A memória é capaz de realizar duas operações, leitura e escrita

1. **leitura** (recebe `1 [org] 1 [end]`): o quarto número da mensagem ([end]) é o endereço da memória que se quer ler. A memória envia uma mensagem de volta tendo como destino a unidade de origem da mensagem que recebeu, como origem a sua própria identificação, como operação um 0 (que indica resposta) e como quarto número o valor encontrado no endereço pedido.
   (envia `[org] 1 0 [valor]`)
2. **escrita** (recebe `1 [org] 2 [end] [val]`): [end] é o endereço onde se quer escrever, [val] é o valor a escrever. A memória substitui o valor que está na posição [end] pelo novo valor fornecido, e envia uma resposta contendo somente um 0.
   (envia `[org] 1 0`)

Unidade de entrada (2)

No nosso caso, vai representar o usuário e seu teclado. Só tem uma operação que o usuário sabe fazer, que é digitar um número; a unidade de entrada envia esse número como resposta.

1. **leitura** (recebe `2 [org] 1`): envia de volta uma mensagem contendo operação 0 e como quarto número o valor digitado pelo usuário.
   (envia `[org] 2 0 [valor]`)

Unidade de saída (3)

Representa o vídeo. Tem uma única operação, que é escrever um valor na tela (que será o quadro).

1. **escrita** (recebe `3 [org] 1 [val]`): o quarto número ([val]) da mensagem recebida é o valor a ser escrito. Escreve esse valor e retorna uma mensagem contendo somente um 0.
   (envia `[org] 3 0`)

Unidade lógica e aritmética (4)

Consegue fazer contas. O quarto e o quinto valores na mensagem ([val1] e [val2]) são os valores a operar. O resultado da operação é enviado de volta como quarto valor em uma mensagem com a operação 0. As operações são:

1. **soma** (recebe `4 [org] 1 [val1] [val2]`): resultado é [val1] + [val2]
2. **subtração** (recebe `4 [org] 2 [val1] [val2]`): resultado é [val1] - [val2]
3. **multiplicação** (recebe `4 [org] 3 [val1] [val2]`): resultado é [val1] × [val2]
4. **divisão** (recebe `4 [org] 4 [val1] [val2]`): resultado é [val1] ÷ [val2]
5. **maior** (recebe `4 [org] 5 [val1] [val2]`): resultado é 1 se [val1] for maior que o [val2], e 0 caso contrário Mensagem de retorno:
   (envia `[org] 4 0 [resultado]`)

Unidade de busca de instruções (5)

Essa unidade contém um número, chamado PC, que inicialmente é zero. Ela sabe fazer duas operações, busca e alteração do PC.

1. **busca** (recebe `5 [org] 1`):
   1. envia uma mensagem para a memória , pedindo para ler o valor no PC
      (envia `1 5 1 [PC]`)
   2. espera o resultado
   3. incrementa o valor no PC (se valia 4, passa a valer 5, por exemplo)
   4. envia uma mensagem de retorno, com o valor lido da memória
      (envia `[org] 5 0 [valor lido]`)
2. **alteração do PC** (recebe `5 [org] 2 [val]`): coloca o [val] da mensagem recebida no PC, envia uma mensagem de retorno
   (envia `[org] 5 0`)

Unidade de decodificação de instruções (6)

Só sabe fazer uma operação, conseguir e decodificar a próxima instrução para a unidade de controle.

1. **decodifica** (recebe `6 [org] 1`):
    1. Busca a próxima instrução (envia `5 6 1`, espera a mensagem de retorno com a instrução)
    2. Encontra a descrição do que deve ser feito para implementar a instrução que tem esse código (ver abaixo)
    3. Entrega essa descrição para a unidade [org] (deve ser a unidade de controle)
    4. Espera o retorno da descrição

Unidade de execução de instruções (7)

É a unidade que vai comandar tudo. Ela vai realizar o que cada instrução disser que tem que ser feito. Esta unidade pede a descrição da próxima instrução para a unidade de decodificação, segue o que está descrito, devolve a descrição para a unidade de decodificação, pede uma nova descrição, e assim por diante, até que a descrição recebida diga que deve parar. Os passos a executar:

1. envia um pedido à unidade de decodificação (envia `6 7 1`)
2. recebe os passos a executar
3. executa esses passos
4. devolve o texto à unidade de decodificação
5. volta ao passo 1

Instrução 1 (ESCREVE)

Esta instrução escreve um número em um dispositivo de saída. Tem duas informações complementares, o endereço onde está o valor a escrever e a identificação do dispositivo. Os passos a executar estão abaixo. Após cada passo que faz um pedido a outra unidade, deve esperar o resultado enviado por ela.

1. pede para a unidade de busca o endereço do valor a escrever
   (envia `5 7 1`)
2. pede para a memória o valor a escrever
   (envia `1 7 1 [valor recebido no passo 1]`)
3. pede para a unidade de busca a identificação do dispositivo
   (envia `5 7 1`)
4. pede para a unidade de saida escrever o resultado
   (envia `[valor recebido em 3] 7 1 [valor recebido em 2]`)

Instrução 2 (LE)

Esta instrução lê um valor de um dispositivo de entrada e coloca o valor lido em uma posição de memória. Tem duas informações complementares, o número do dispositivo e o endereço de memória onde colocar o valor. Os passos para sua execução estão abaixo. Após cada passo que faz um pedido a outra unidade, deve esperar o resultado enviado por ela.

1. pede para a unidade de busca a identificação do dispositivo
   (envia `5 7 1`)
2. pede para a unidade de entrada o próximo valor
   (envia `[valor recebido em 1] 7 1`)
3. pede para a unidade de busca o endereço onde colocar o valor lido
   (envia `5 7 1`)
4. pede para a memória guardar o valor
   (envia `1 7 2 [valor recebido em 3] [valor recebido em 2]`)

Instrução 3 (SOMA)

Esta instrução realiza a soma de dois valores. Tem 3 informações complementares, os endereços dos dois valores a somar e o endereço onde deve colocar o resultado. Os passos para sua execução estão abaixo. Após cada passo que faz um pedido a outra unidade, deve esperar o resultado enviado por ela.

1. pede para a unidade de busca o endereço onde está o primeiro valor
   (envia `5 7 1`)
2. pede para a memória o primeiro valor
   (envia `1 7 1 [valor recebido no passo 1]`)
3. pede para a unidade de busca o endereço onde está o segundo valor
   (envia `5 7 1`)
4. pede para a memória o segundo valor
   (envia `1 7 1 [valor recebido no passo 3]`)
5. pede para a ULA fazer a soma
   (envia `4 7 1 [valor do passo 2] [valor do passo 4]`)
6. pede para a unidade de busca o endereço onde colocar o resultado
   (envia `5 7 1`)
7. pede para a memória guardar o valor
   (envia `1 7 2 [valor do passo 6] [valor do passo 5]`)

Instrução 7 (DESVIO INCONDICIONAL)

Esta instrução causa o desvio da execução: as próximas instruções devem passar a ser buscadas em outro lugar. Tem 1 informação complementar, o endereço da instrução para onde deve desviar. Os passos para sua execução estão abaixo. Após cada passo que faz um pedido a outra unidade, deve esperar o resultado enviado por ela.

1. pede para a unidade de busca o endereço onde está a próxima instrução
   (envia `5 7 1`)
2. realiza o desvio, pedindo para a unidade de busca alterar o valor do PC
   (envia `5 7 2 [valor recebido do passo 1]`)

Instrução 4 (SUBTRAI)

Esta instrução realiza a subtração de dois valores. Tem 3 informações complementares, os endereços dos dois valores a subtrair e o endereço onde deve colocar o resultado. Os passos para sua execução estão abaixo. Após cada passo que faz um pedido a outra unidade, deve esperar o resultado enviado por ela.

1. pede para a unidade de busca o endereço onde está o primeiro valor
   (envia `5 7 1`)
2. pede para a memória o primeiro valor
   (envia `1 7 1 [valor recebido no passo 1]`)
3. pede para a unidade de busca o endereço onde está o segundo valor
   (envia `5 7 1`)
4. pede para a memória o segundo valor
   (envia `1 7 1 [valor recebido no passo 3]`)
5. pede para a ULA fazer a subtração
   (envia `4 7 2 [valor do passo 2] [valor do passo 4]`)
6. pede para a unidade de busca o endereço onde colocar o resultado
   (envia `5 7 1`)
7. pede para a memória guardar o valor
   (envia `1 7 2 [valor do passo 6] [valor do passo 5]`)

Instrução 5 (DIVIDE)

Esta instrução realiza a divisão de dois valores. Tem 3 informações complementares, os endereços dos dois valores a dividir e o endereço onde deve colocar o resultado. Os passos para sua execução estão abaixo. Após cada passo que faz um pedido a outra unidade, deve esperar o resultado enviado por ela.

1. pede para a unidade de busca o endereço onde está o primeiro valor
   (envia `5 7 1`)
2. pede para a memória o primeiro valor
   (envia `1 7 1 [valor recebido no passo 1]`)
3. pede para a unidade de busca o endereço onde está o segundo valor
   (envia `5 7 1`)
4. pede para a memória o segundo valor
   (envia `1 7 1 [valor recebido no passo 3]`)
5. pede para a ULA fazer a divisão
   (envia `4 7 4 [valor do passo 2] [valor do passo 4]`)
6. pede para a unidade de busca o endereço onde colocar o resultado
   (envia `5 7 1`)
7. pede para a memória guardar o valor
   (envia `1 7 2 [valor do passo 6] [valor do passo 5]`)

Instrução 8 (DESVIO CONDICIONAL MAIOR)

Esta instrução causa o desvio da execução, condicionado à comparação de dois valores: se o primeiro for maior que o segundo, realiza o desvio, senão não desvia. Tem 3 informações complementares, os endereços dos dois valores a comparar e o endereço da instrução para onde deve desviar. Os passos para sua execução estão abaixo. Após cada passo que faz um pedido a outra unidade, deve esperar o resultado enviado por ela.

1. pede para a unidade de busca o endereço onde está o primeiro valor
   (envia `5 7 1`)
2. pede para a memória o primeiro valor
   (envia `1 7 1 [valor recebido no passo 1]`)
3. pede para a unidade de busca o endereço onde está o segundo valor
   (envia `5 7 1`)
4. pede para a memória o segundo valor
   (envia `1 7 1 [valor recebido no passo 3]`)
5. pede para a ULA fazer a comparação
   (envia `4 7 5 [valor do passo 2] [valor do passo 4]`)
6. pede para a unidade de busca o endereço para onde desviar
   (envia `5 7 1`)
7. se o valor recebido da ULA (passo 5) for 1, faz o desvio; se não, não faz mais nada nesta instrução. Para fazer o desvio, pede para a unidade de busca alterar o valor do PC
   (se for 1, envia `5 7 2 [valor do passo 6]`)

Instrução 9 (PARA)

Essa instrução causa a parada da CPU. Os passos a executar são os seguintes:

1. Para

Programa 1

.	.	.	.	.	.	.	.	.	.
endereço	1	2	3	4	5	6	7	8	9
valor	2	20	2	2	21	3	20	21	22
endereço	11	12	13	14	15	16	17	18	.
valor	22	18	23	1	23	3	9	2	.

Programa 2

.	.	.	.	.	.	.	.	.	.
endereço	1	2	3	4	5	6	7	8	9
valor	2	18	2	2	19	8	18	19	14
endereço	11	12	13	14	15	16	17	18	19
valor	19	20	18	1	18	3	9	0	0
endereço	20	.	.	.	.	.	.	.	.
valor	0	.	.	.	.	.	.	.	.

Programa 3

.	.	.	.	.	.	.	.	.	.
endereço	1	2	3	4	5	6	7	8	9
valor	12	3	3	12	13	12	8	14	12
endereço	11	12	13	14	.	.	.	.	.
valor	9	1	1	6	.	.	.	.	.

Programa 4

.	.	.	.	.	.	.	.	.	.
endereço	1	2	3	4	5	6	7	8	9
valor	2	15	1	15	3	3	15	16	15
endereço	11	12	13	14	15	16	17	.	.
valor	15	17	3	9	0	-1	0	.	.