**Descrição do processo de “ADD ax,bx”:**

- Todo processo precisa ser escrito na memoria para que possa ser executado. Sendo assim, uma vez que o processo já foi escrito, precisamos buscar o processo em memória para que, em um segundo momento, possamos pegar esse processo e executa-lo.

**Descrição do processo de busca na memória (circuito não minimizado):**

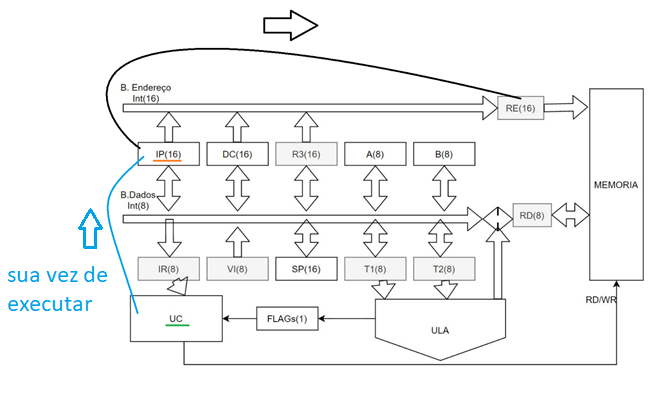
- A cada instrução dada pela unidade de controle (UC) contamos como 1 clock

**Clock 1:**

IP = Instruction Pointer

RE = registrador de endereços

O IP aponta para a próxima instrução a ser executada.

ip --(barramento de enredo)--> Registro de endereços(RE)

**clock 2:**

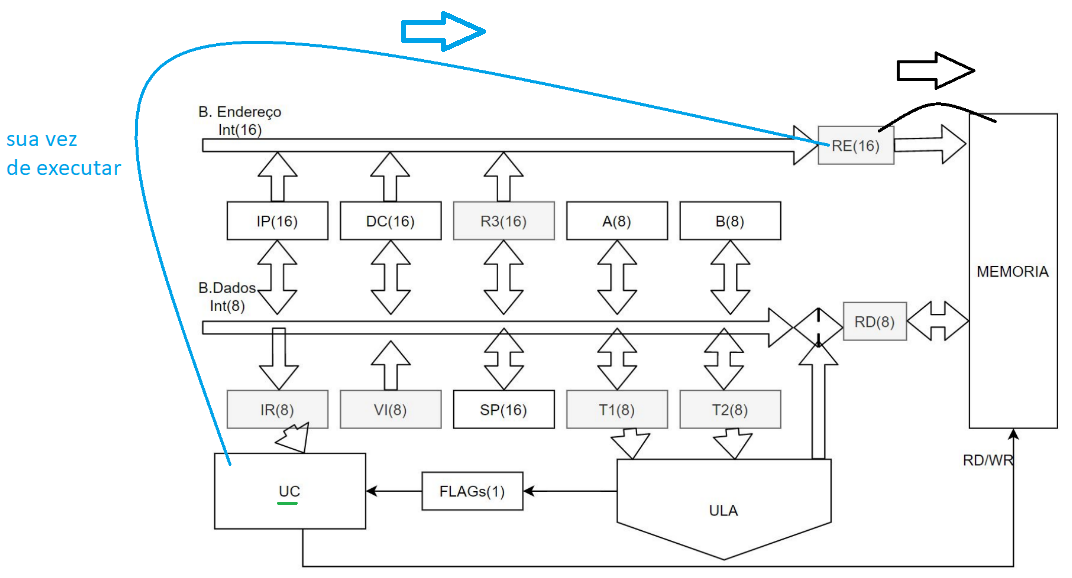
o registrador de endereços possui o endereço de memória que estamos procurando, no caso, o RE possui o endereço 0x0053. Com o endereço 0x0053, consultamos na memória esse processo para que possamos busca-lo e, posteriormente, executar.

2.0) registro de endereços (RE) 🡪 memoria (para leitura)

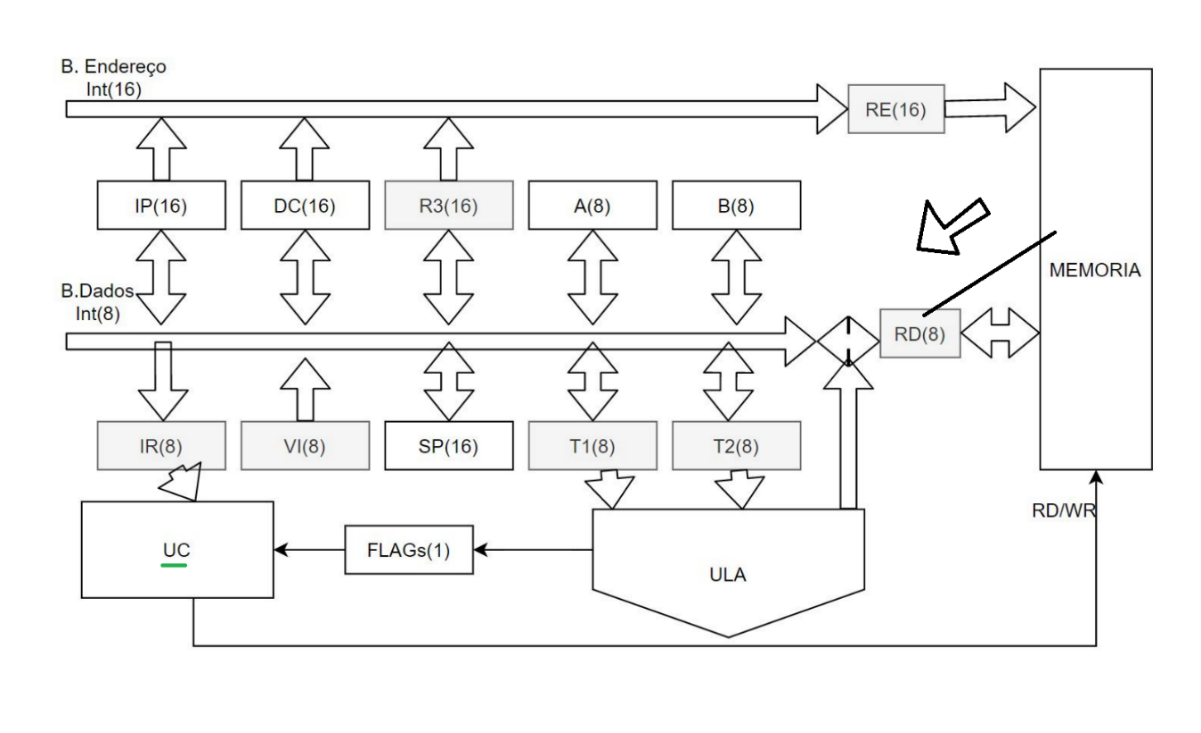
2.1) memoria(leitura) 🡪 registro de dados (RD)

Em 2.1 passamos para da memória para RD o CONTEÚDO que havia no endereço 0x0053

**ilustração do 2.0:**



**Ilustração do 2.1:**



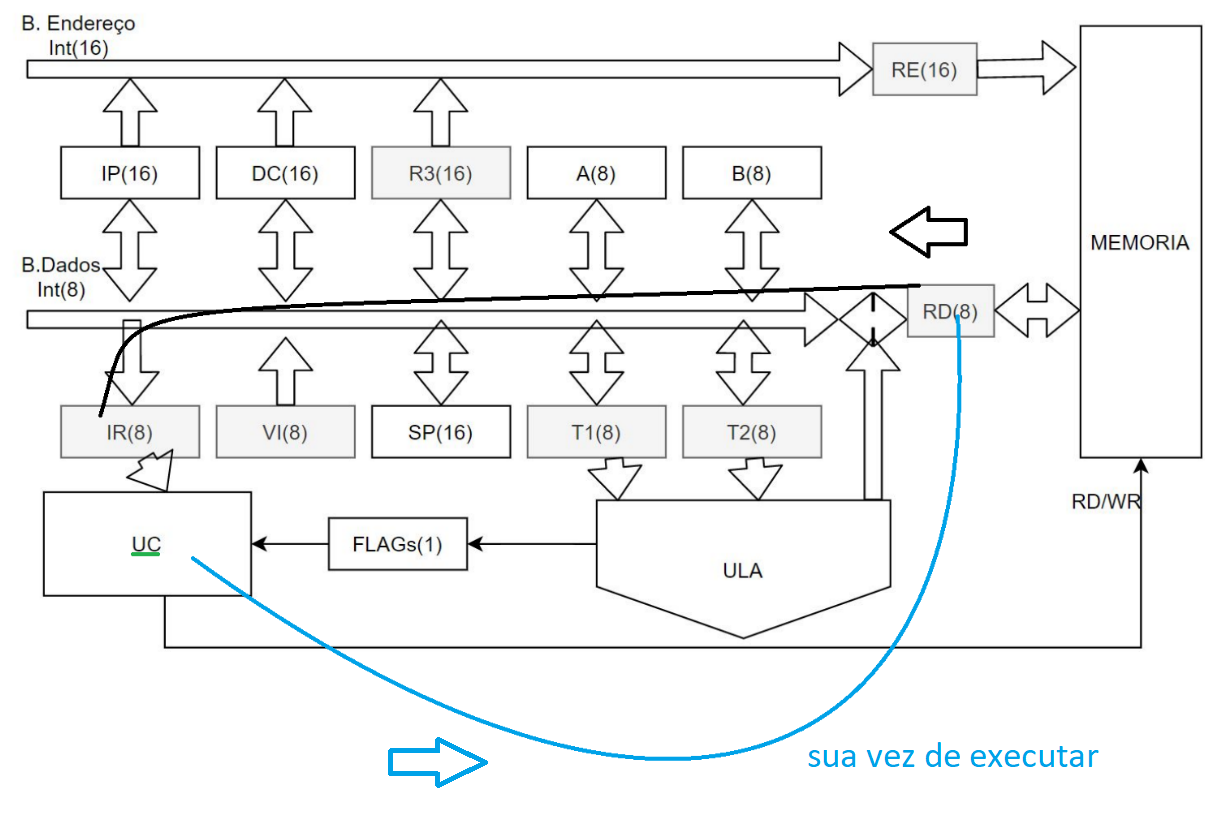
**clock 3:**

“pausa de 1 clock”, não é exatamente um clock, mas abstraímos para facilitar a explicação.

**clock 4:**

IR = Instruction Register = registro de instruções

registro de dados (RD) ---(barramento de dados)--> registro de instruções (IR)



clock 5:

- A instrução vai para do registro de instruções (IR) 🡪 unidade de controle (UC)

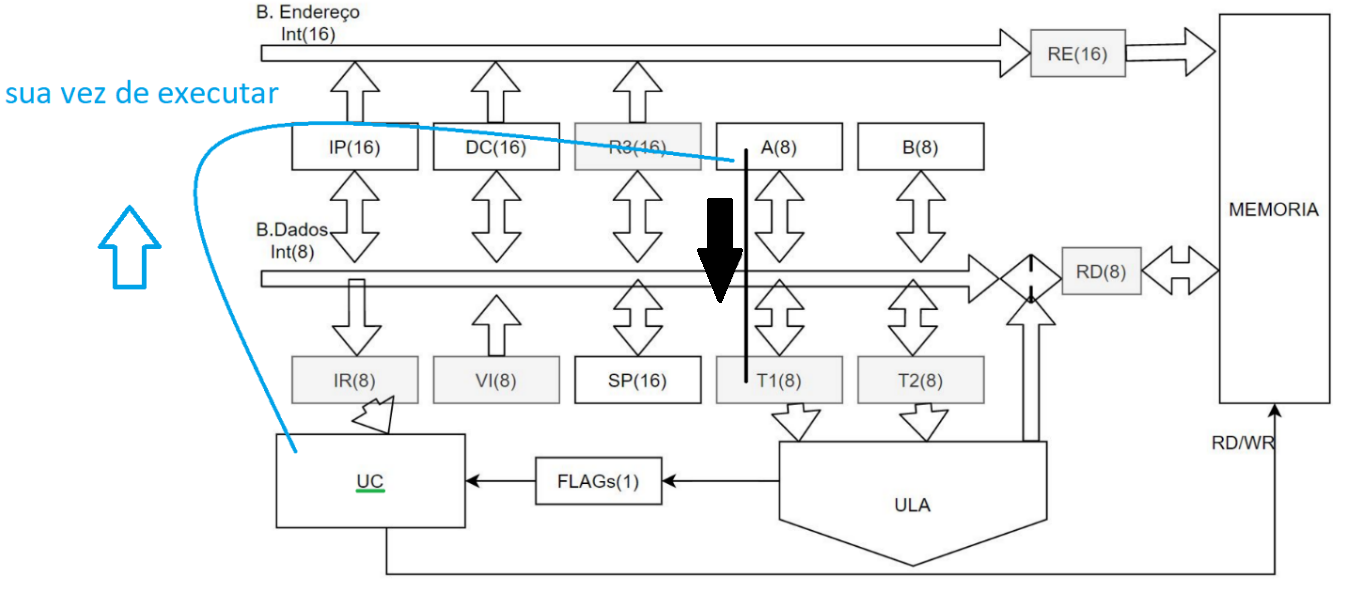
**Descrição do processo de execução (circuito não minimizado):**

- Uma vez que a unidade de controle possui o processo, “notifica” os registradores para realizar a tarefa necessárias para execução do comando especificado.

- O barramento de dados pode carregar apenas 1 informação por vez.

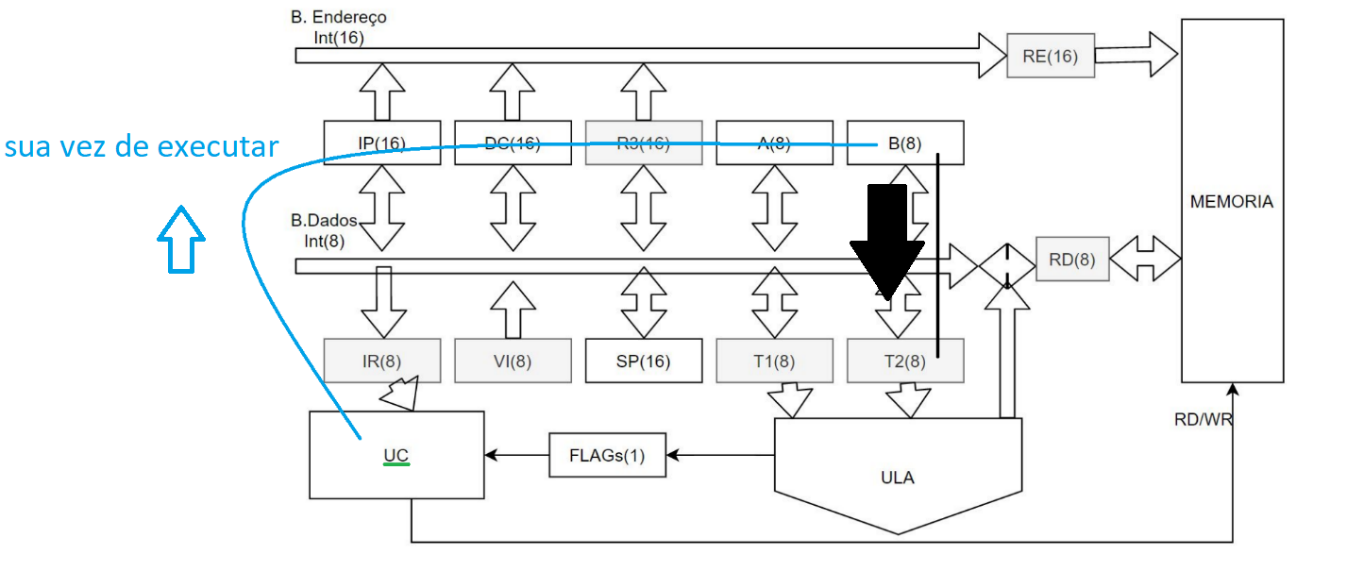
**Clock 6:**

A ----(barramento de dados)---> t1



**Clock 7:**

B ----(barramento de dados)---> t2



**Comportamento do t1, t2 e unidade lógica aritmética:**

uma vez que o dado chega no t1, ele fica na “porta de entrada” da ULA esperando o próximo dado chegar.

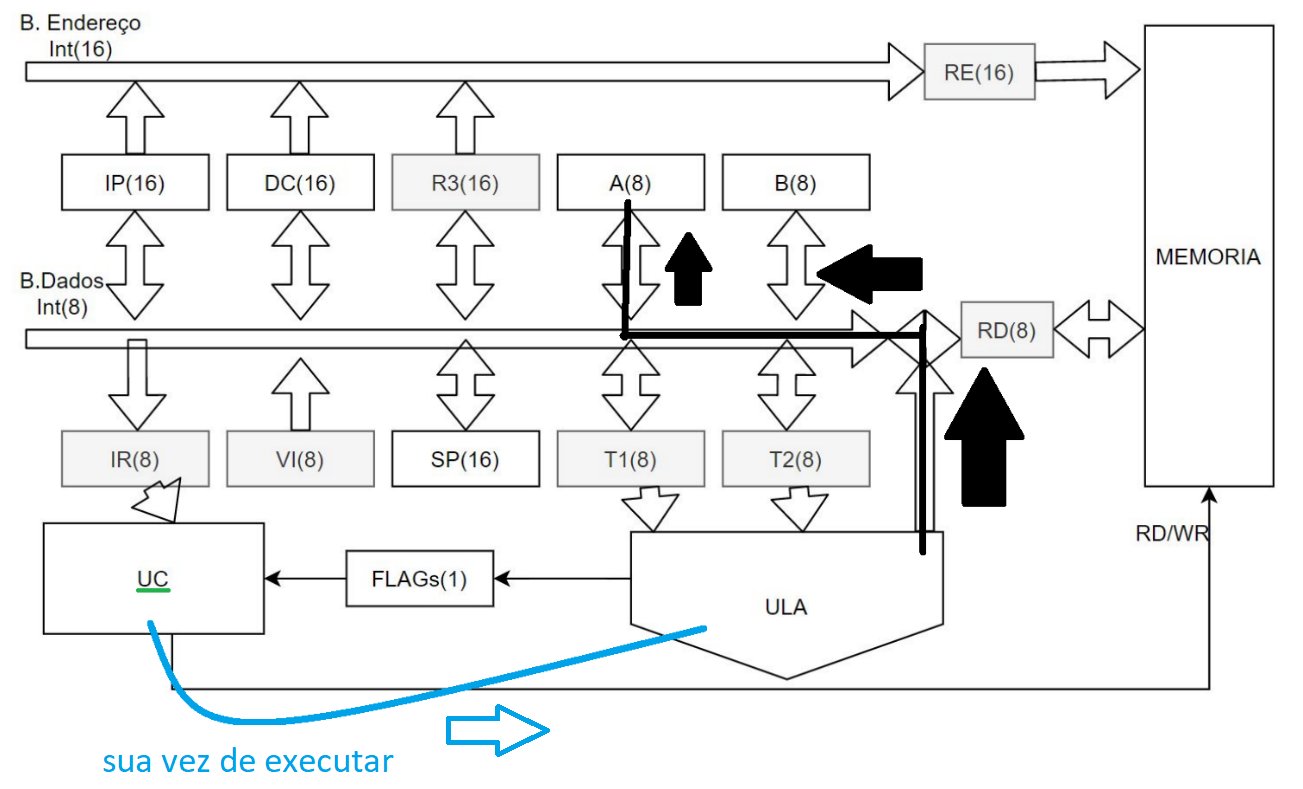
Uma vez que ambos os dados de t1 e t2 chegam, os dois estão localizados na entrada da unidade lógica aritmética (ULA), basta realizar a operação desejada e, para realizar essa operação é tempo constante, por isso não é necessário esperar um outro clock.

Assim, quando t1 e t2 estão na entrada, produzimos quase que instantaneamente a saída, com base na operação deseja, no caso da tag, somamos ax e bx. Essa operação de soma é realizada por uma flag especifica.

**Clock 8:**

- precisamos levar a saída para ser armazenada em algum lugar, no caso da tag, desejamos fazer ADD ax, bx , ou seja, o valor de ax = (soma de ax,bx)

Saída da ULA ----(barramento de dados)-----> A



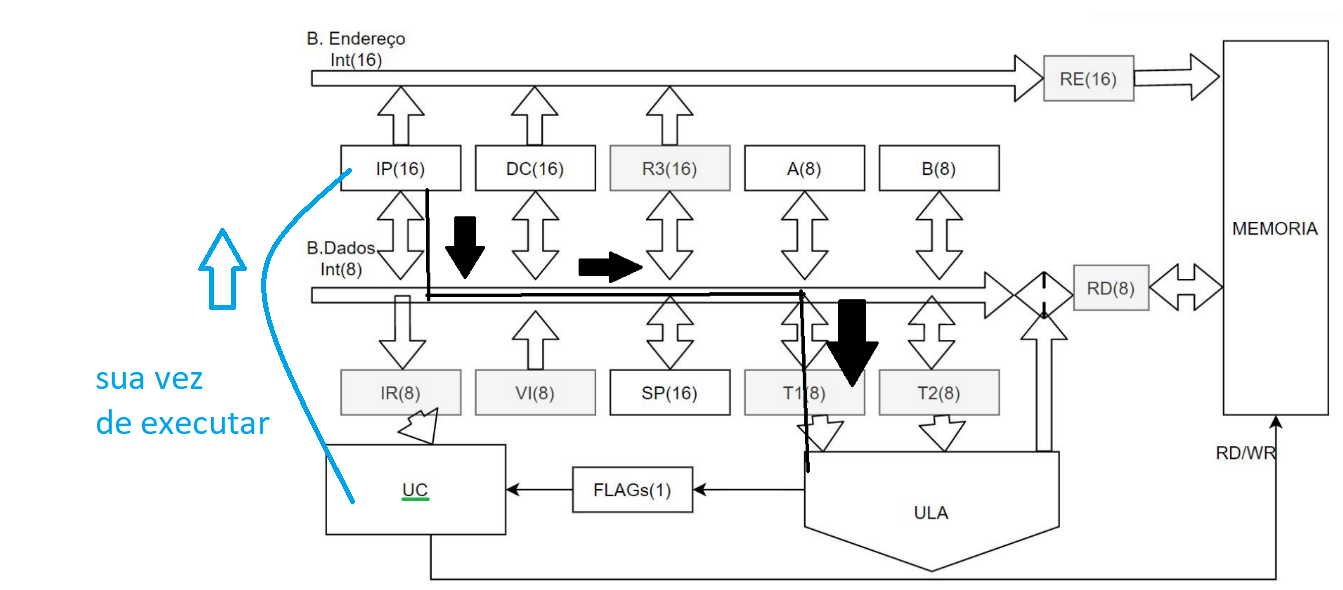
**Clock 9**

Barramento de dados aceita 8 bits

IP low -----(barramento de dados)-----> t1

Para o t1 carregamos os bits menos significativos de 0x0053, ou seja, carregamos o “0x53”

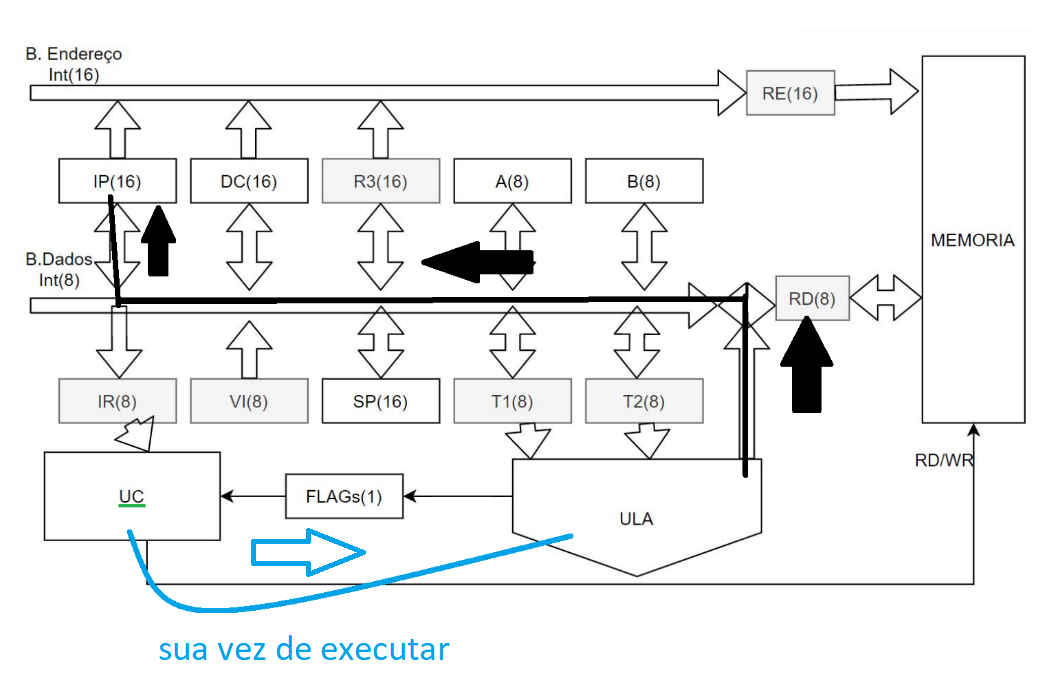
Com o 0x53 na ENTRADA da ULA, somamos +1 em 0x53



**Clock 10:**

- Enviamos o 0x54 de volta para o IP-low

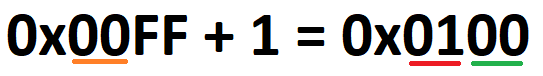
Saída da ULA ----(barramento de dados)----> IP-low



**Alterações das instruções em caso de carry:**

Havendo carry ao somar +1 em “clock 9”, precisamos ajustar o nosso número, modificando-o.

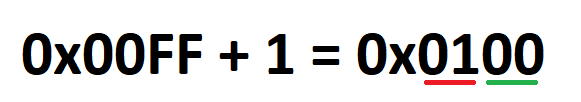
Exemplo em que há carry:



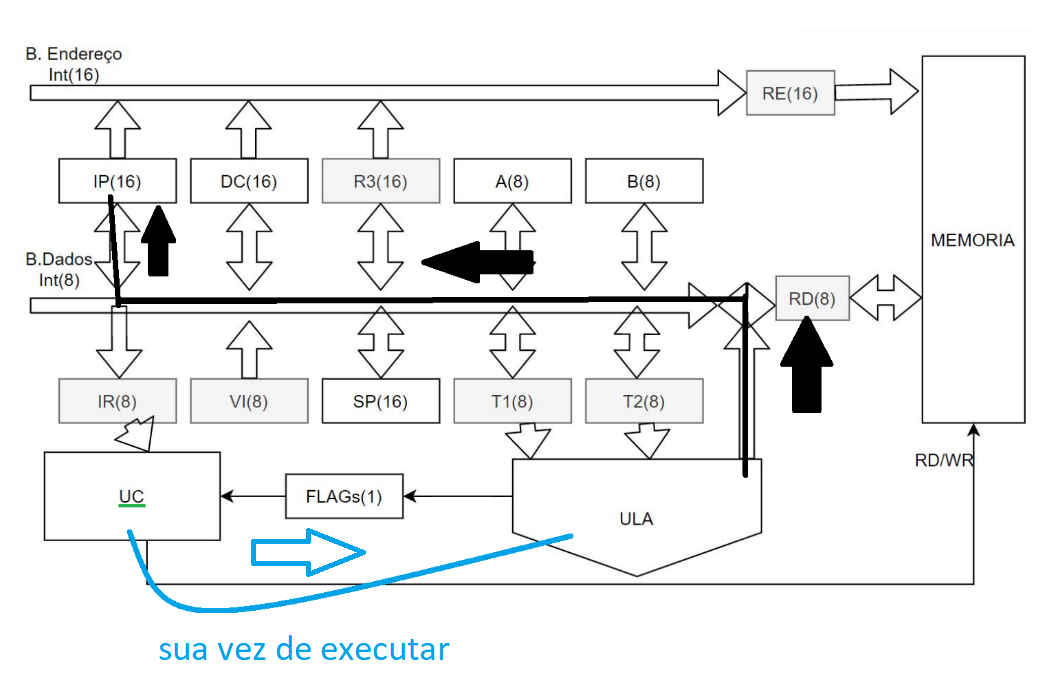
Precisamos carregar a parte em verde para de volta ao ip-low(responsável pelos 8 bits menos significativos), e mandar para a ULA a parte do ip-high (em laranja)

**Alterações no clock 10:**

- No exemplo dado, enviamos o “0x00” (sublinhado em verde) de volta para o IP-low (responsável pelos 8 bits menos significamos)

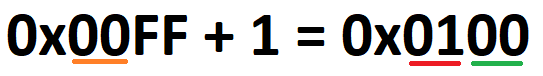


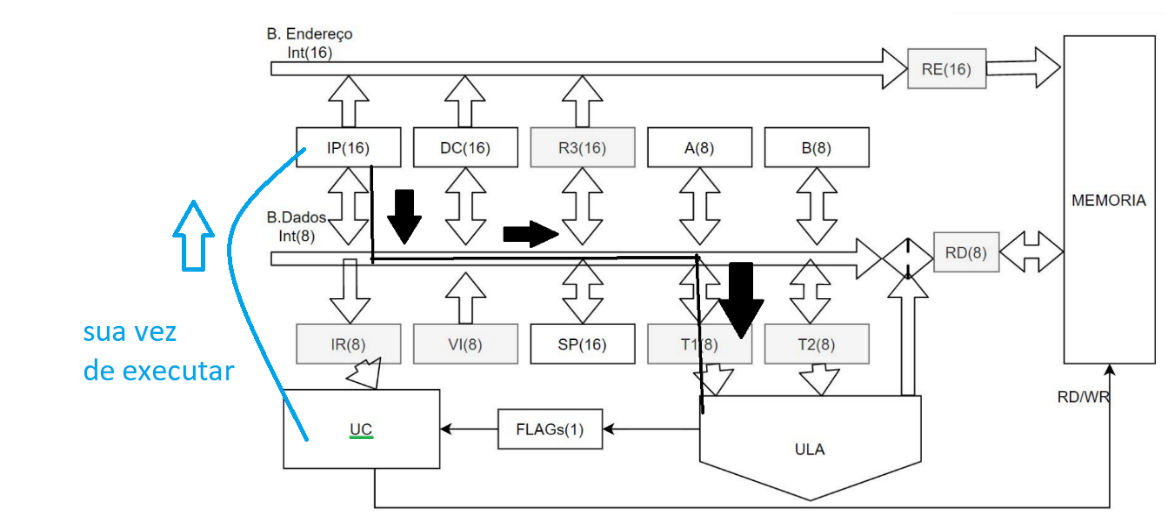
Saída da ULA ------(barramento de dados)-----> IP-low



**Primeiro clock extra (clock 11):**

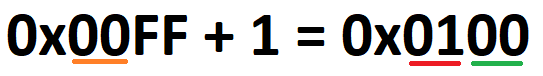
- é preciso somar +1 na parte sublinhada de laranja(por conta do carry), então enviamos essa parte laranja do ip high para a ULA



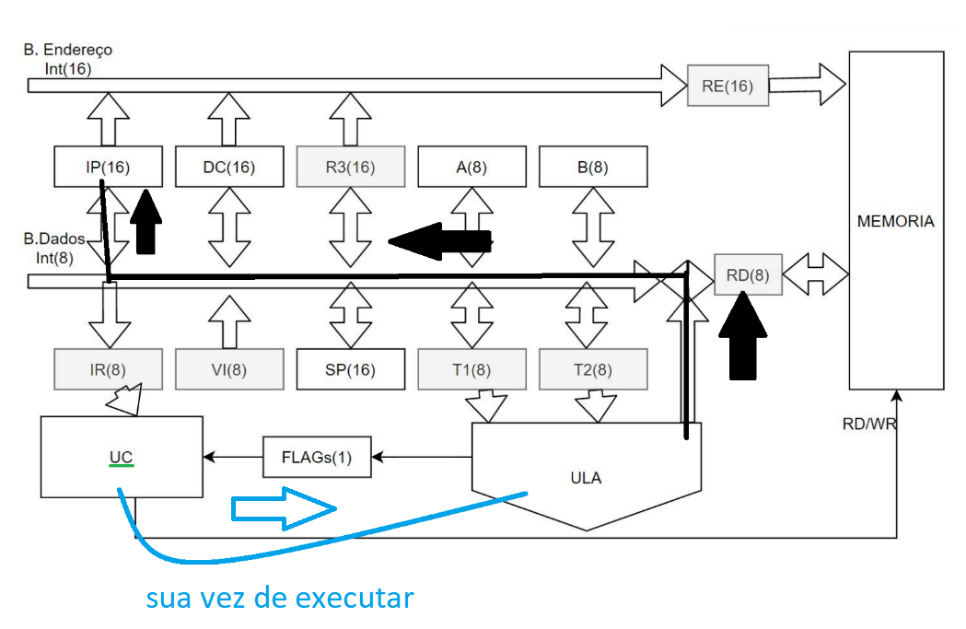
IP high ----(barramento de dados)----> ULA

**segundo clock extra(clock 12):**

- Após a soma de +1 feita pela ULA, enviamos o “0x01” sublinhado em vermelho de volta para o IP-high



Saída da ULA ------(barramento de dados) -----> IP high



**Optimização do circuito:**

Durante o (clock 3), foi dito que houve uma pausa. Podemos aproveitar esse momento de desuso do nosso circuito para realizar tarefas e otimizar o sistema.

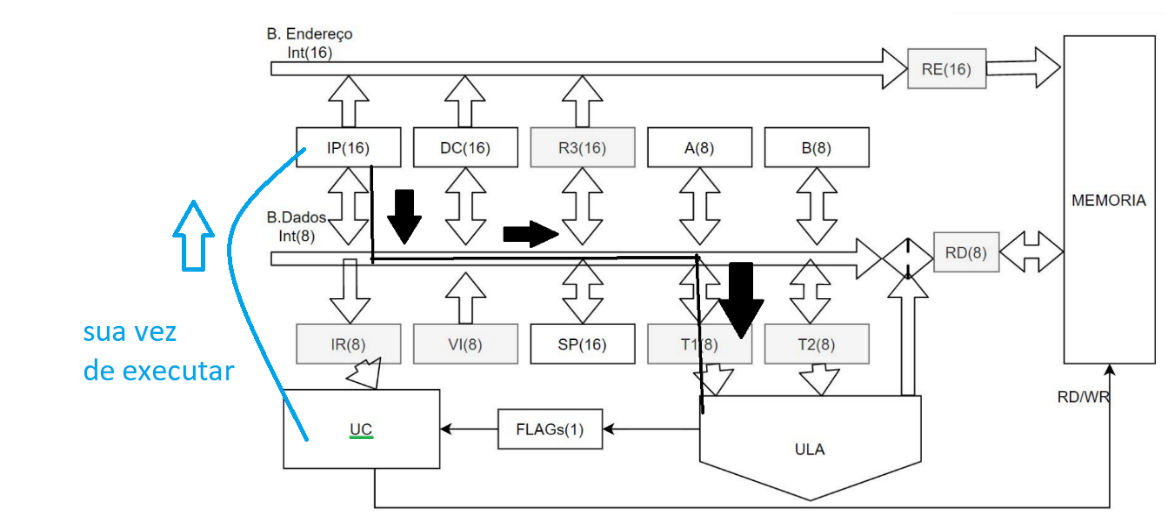
No clock 3 podemos fazer as operações do clock 9, e quando chegar a fez de realizarmos o clock 9, essa tarefa já vai ter sido feita anteriormente.

**Clock 9**

Barramento de dados aceita 8 bits

IP low -----(barramento de dados)-----> t1

Para o t1 carregamos os bits menos significativos de 0x0053, ou seja, carregamos o “0x53”

Com o 0x53 na entrada da ULA, somamos +1 em 0x53