

```

push H [H]           push C [D]
push G [H]           add    [B+C]
push F [G,H]         push B [C]
add    [D+H]          add    [(B+C)*D+E]
push E [F]           push A [F]
add    [E+F]          add    [G*H]
push D [G,F]         null   [B+C]*D+E+F
add    [G*H]          null   [G*H]
A = (B-C)*(H+D)

```

```

push D [B]           add    [B-C]
push H [H]           add    [H+D]
add    [H+D]          div    [B-C]
push C [H+D]         push A [H+D]
push B [C]           (MUL)  [H+D]

```

Observação Cache é a memória do processador que serve para processar mais rapidamente informações. Ela armazena informações muito utilizadas para que seja mais rápida sua execução.

A memória cache pode ser apresentada da seguinte forma:

$$L_1, L_2, L_3 \text{ e } L_4 \quad L = \text{Level}$$

Quanto maior o L , menor a quantidade de memória, porém, mais rápida é a execução.

A cache pode variar de 512KB a 16MB.

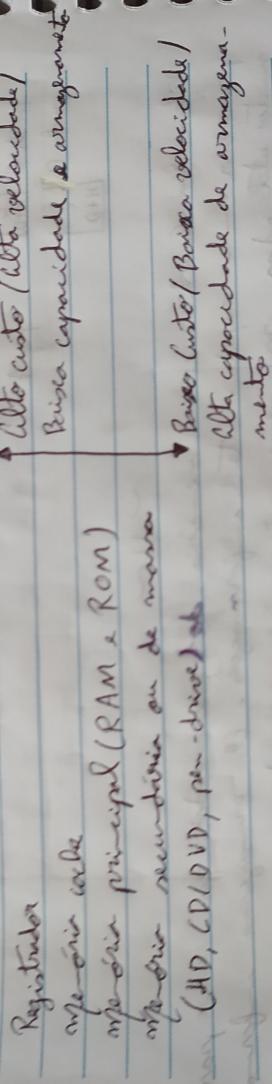
Write-through: sempre que um dado é atualizado na cache ele é atualizado na ram logo depois.

kajoma

Wait - Halt + hots - ligados não ocorre de imediato, ele primeiro ativaiza o bloco da cache e só deixa que o dado precise ser atualizado através de uma hit de controle. Ele será atualizado na memória quando precisar alterar o bloco.

Centro de Unite - Busk; caso se utilize um sistema de DMA (Direct Memory Access) (acesso direcionado a interface da RAM) pode ser que ele recorra um dado desatualizado da RAM pois ele só atualizará onde da cache.

Hierarquia de memória



- Princípio de localidade espacial
"De que bloco nova instrução, os chances de seu gerer ficar a instrução seguinte é alta". Especialmente, na memória, os instruções próximas estarão em posições contíguas.
- Poluição: colocar no cache um bloco da memória RAM.

- Princípio da localidade temporal
"De em linguagem uma palavra agora, a chance de ser usada novamente em um curto período de tempo é alta".
- Poluição: armazenar dados recorrentes em uma memória mais rápida (cache).

Eficiência de memória cache:

- Hit (acerto):
 - Cadeia guarda o processador busca uma "palavra" da cache e a encontra.
- Miss (fallha):
 - Cadeia guarda o processador busca uma "palavra" da cache e não a encontra.
 - Taxa de acerto:
 - $T_a = (N_h) / (N_h + N_f)$
 - Ta → Taxa de acertos
 - Nh → Número de acertos
 - Nf → Número de falhas

Substituição em cache

- FIFO (first-in, first-out): funcionamento de fila, primeiro a entrar é o ultimo a sair.
- LRU (least recently used): a linda a ser substituída é a que está há mais tempo sem ser utilizada. São utilizados lists associados aos blocos da cache, indicando o instante de tempo em que o bloco entrou.
- Clockwise: Quando a cache encontra uma posição e liberado imediatamente.

Mapamento associativo

Não há prévisão para onde vai ser alojado na cache o bloco retirado da Ram. A substituição é feita através dos métodos descritos em cima.

Mapamento direto

O processador manda pra qual linha da cache ele deve levar o bloco da memória RAM.

Conferir com o Walter: Bloco na cache = bloco da RAM mod número de linhas

REDEMI NOTE 8 48MP QUAD CAMERA

● O

Momentaneamente associado ao conjunto

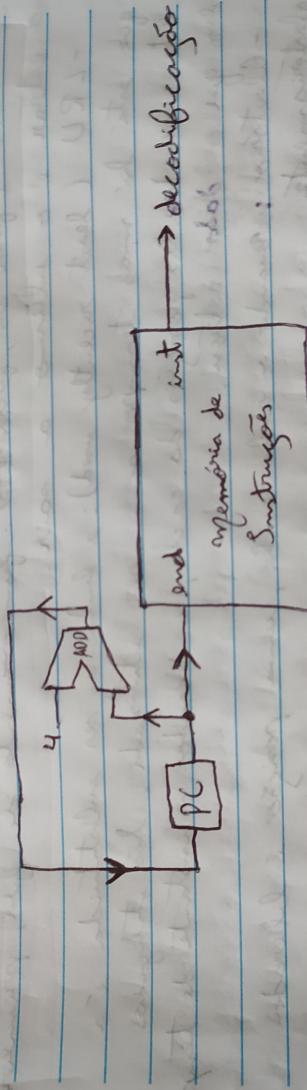
a memória cache será dividida por conjuntos e o posicionamento um bloco na cache, ela será aderada no bloco correspondente da cache que estiver vazio.

~ ~ ~

Ciclo de instruções

Bloco de Instruções

Realiza uma busca de instruções, fazer um cálculo da próxima instrução para ser executada e fornecer essa instrução para a próxima fase: a decodificação.



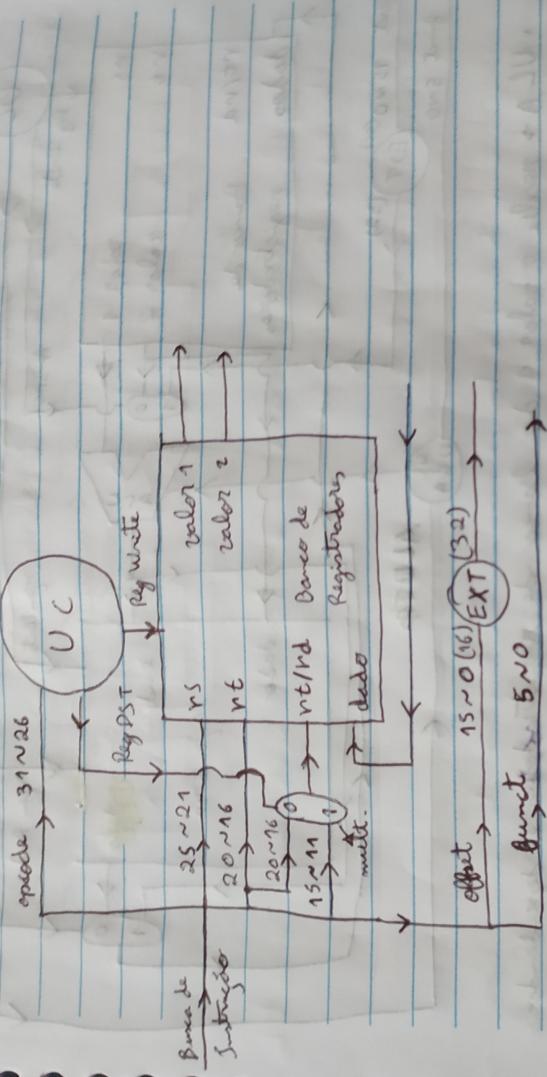
• PC → Program Counter → indica qual será a próxima instrução a ser lida.

• ADP → Una UAL (Unidade lógica aritmética) / dispositivo que realiza operações lógicas e aritméticas) que não realiza soma, faz subtração e增值 o valor do PC atual e adiciona mais 4.

• Memória de Instruções → Fornecê uma sequência de instruções de 4 em 4 bits que serão identificados por um endereço e que retorna a instrução a ser executada e endereçada encontradas, e enviando para a decodificação.

Decodificação de Instruções e Busca nos Registradores
 Realiza a decodificação de acordo com o formato da instrução que está sendo executada (jmp, add, sub, etc.).

opcode 31~26



• opcode → envia dado (31~26) para a unidade de controle, auxiliando no processamento e no controle de ciclo de instruções.

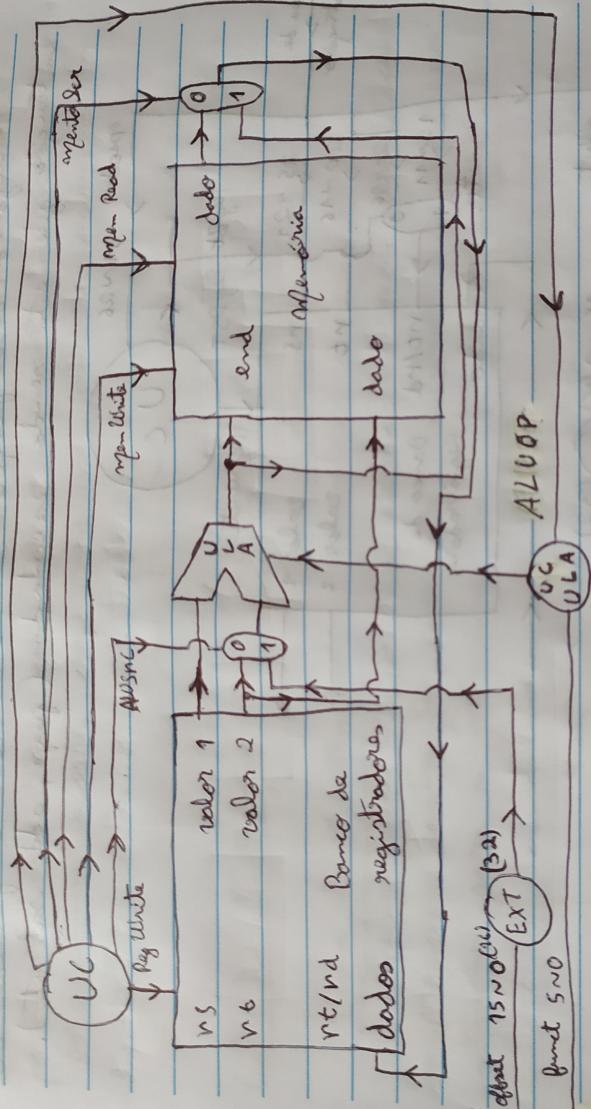
• Banco de registradores → fornece um registrador de origem (rs), um registrador temporário (rt), um registrador temporário ou registrador de destino (rd) para varrer dependendo do nível da UC e novo dado a ser escrito.

• Multi. (Multiplicador), escolhe se será usado o multiplicador temporário ou o registrador de destino. Esse multiplicador define os instruções de tipo v para as do tipo i (tipo v só se de operações aritméticas).

• offset → é um endereço de endereço que sera enviado para 32 bits e sera usado depois em outra etapa.
 função → caso esteja executando um jmp ele irá para ~~lugar certo~~

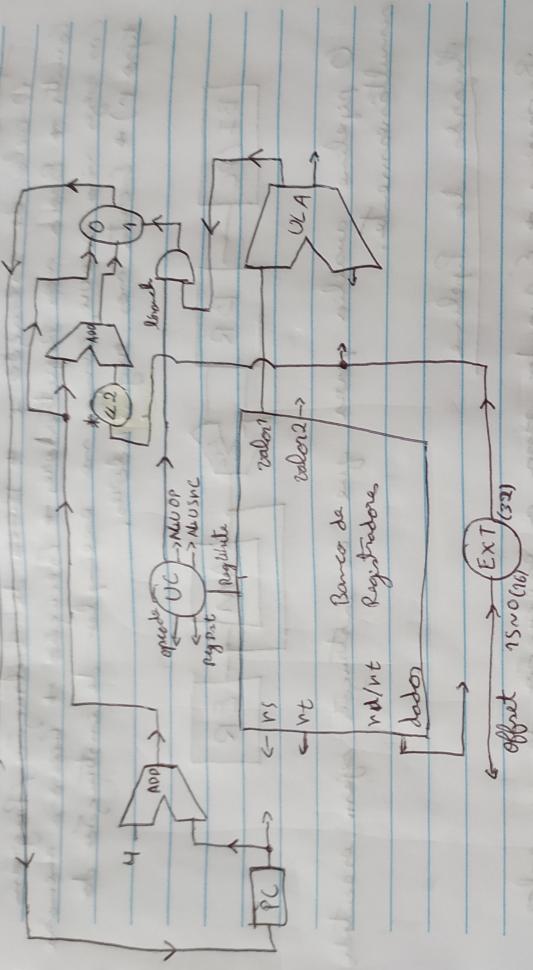
B S T Q Q S S

Evolução, basta a Memória e basta nos Registradores.



- UC → recebe o valor 1 de banco de registradores e depende do sinal recebido pela UC, ele em recebe o valor 2 em ex offset. A operação realizada era dependente da UC ULA (que recebe a função ALUOP, que sai da UC). Após a operação o resultado irá em para a memória ou para o multiplexador que fica operando a memória. Esse multiplexador era responsável por batalhar selecionado para os dados do Banco de Registradores.
- Open Write → realiza a escrita de memória.
- Open Read → realiza a leitura de uma informação da memória.

Ejecución de Branch (BEQ) - TI P0 T

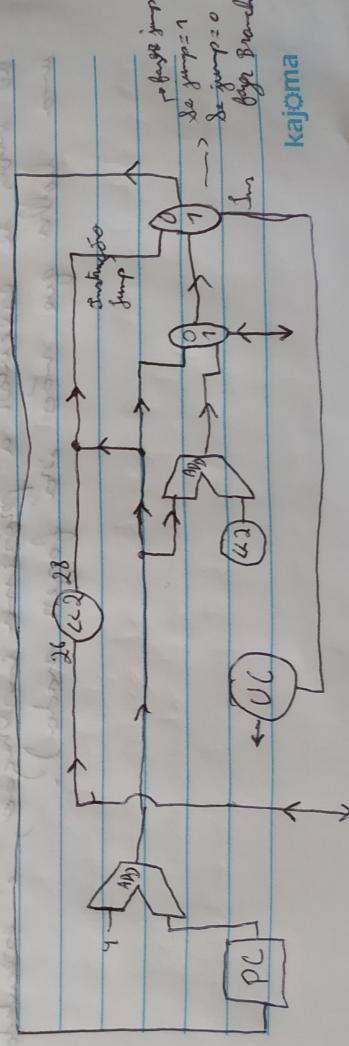


Beq mó^o trabalha com dentro a memória e não com o offset sendo adicionado com o pc + 4, sendo encodado através de um multiplicador.

* Offset definido 2 bits para obter um endereço menor

Ejecución de jump

Se adiciona mais um multiplicador no qual recebe informação da UC para decidir se ira o jump ou não.



Pipeline

O ciclo de execução é → Busca de Instruções / Program Counter → Busca nos Registradores → Execução → Criação e memória (as vezes não ocorre, como no exemplo de instruções do tipo r) → Criação nos registradores.



O pipeline auxilia para que as instruções sejam executadas simultaneamente.

Observações de Pipeline

- Bypassing → É quando para a execução de um valor de uma instrução para a próxima (ocorre por exemplo em uma instrução de add, no qual antes mesmo do registrador ser escrito no bloco de registradores, ele já é enviado com seu novo valor depois da parte de execução).
- Stall (Bubbles) → É quando a instrução precisa esperar que sua determinada fase seja concluída para que ela seja executada, ficando em um estado de espera (ocorre por exemplo em uma instrução de LW, na qual o registrador só terá valor na parte de escrita dos registradores ao invés de executar a próxima instrução, é executado um bloco que não irá fazer nada).