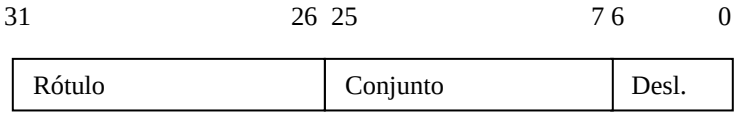
UNIVERSIDADE FEDERAL FLUMINENSE
INSTITUTO DE COMPUTAÇÃO
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

TCC00165-Fundamentos de Arquiteturas de Computadores – Turma :A1
Gabarito - Lista 2

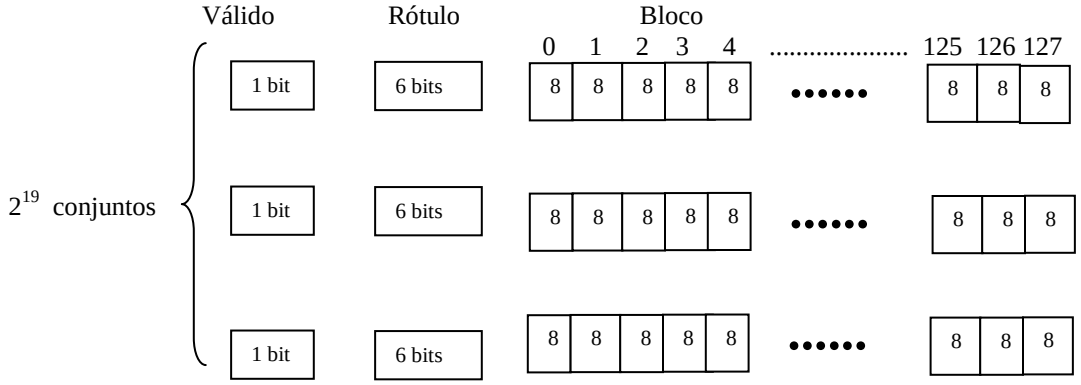
1.
   a) 64M-1= 67108863
   b) Log_2 64M= Log $_2 2^{26}$=26 bits
   c) O barramento de dados deve ter 16 bits.
   d) Número de células × bits/célula = 64M X 16 = 1024 Mbits= 1G bits

2. Considerações gerais:
   Como a máquina pode endereçar 4 Gbytes e cada endereço acessa um byte, temos 4G células. Para endereçar 4G células, precisamos de 32 bits. Logo um endereço da memória principal possui 32 bits. A memória cache pode armazenar 512K blocos, logo ela possui 512K linhas. Como cada bloco que é transferido entre a memória principal e a cache possui 128 bytes, temos que os 7 bits menos significativos do endereço serão utilizados para identificar o byte que se quer dentro de um bloco, em qualquer dos mapeamentos.
   a) mapeamento direto
      Neste caso, temos uma linha por conjunto, logo teremos 512K conjuntos. Em relação aos bits do endereço, precisamos de 7 bits para indicar o byte dentro do bloco, 19 bits para indicar o conjunto dentro da cache e 6 bits para indicar o rótulo (tag).

      Endereço

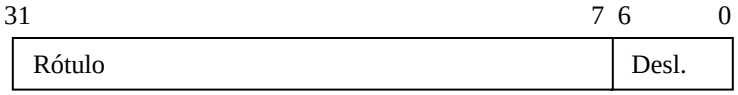


      Memória cache:

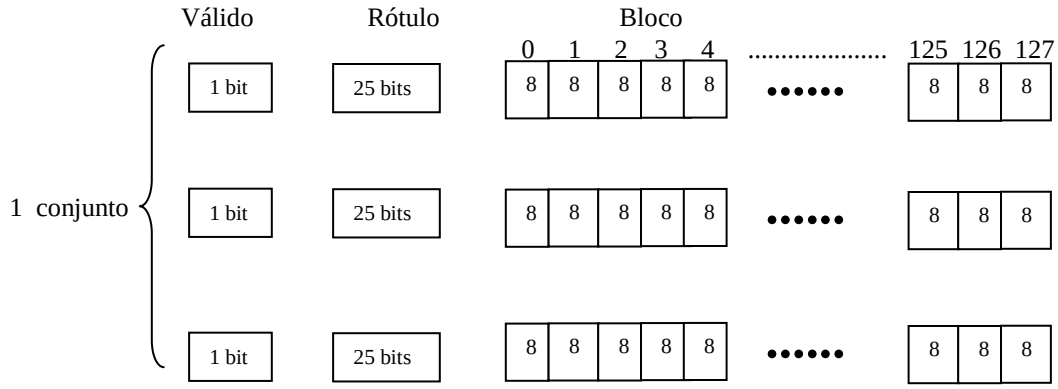


   b) mapeamento totalmente associativo
      Neste caso, temos um único conjunto. Logo, em relação aos bits do endereço, precisamos de 7 bits para indicar o byte dentro do bloco e 25 bits para indicar o rótulo (tag).

Endereço

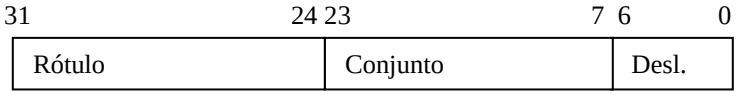


Memória cache:

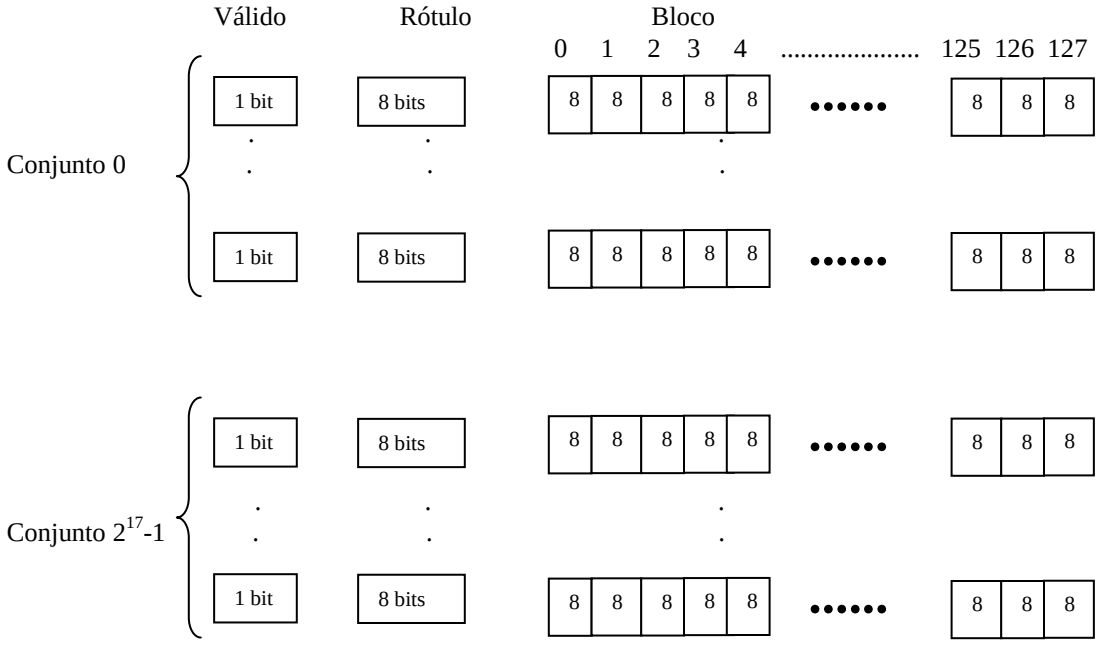


c) mapeamento associativo por conjunto com 4 linhas por conjunto.
Neste caso, temos 4 linhas por conjunto, então teremos 512K/4=128K conjuntos. Logo, em relação aos bits do endereço, precisamos de 7 bits para indicar o byte dentro do bloco, 17 bits para indicar o conjunto dentro da cache e 8 bits para indicar o rótulo (tag).

Endereço:



Memória cache:



3.

a) Tempo de acerto = 1ns
Penalidade por falta = 10 × 1ns = 10ns
Taxa de faltas = 0.01
Logo TMAM = 1 + 0.01 × 10ns = 1.1ns

b) Tempo de acerto = 1.2 × 1ns = 1.2 ns
Penalidade por falta = 10 × 1ns = 10ns
Taxa de faltas = 0.005
Logo TMAM = 1.2 + 0.005 × 10ns = 1.25 ns
Então esta mudança não irá melhorar o desempenho da máquina pois o TMAM aumentou com estas mudanças.

4.

a) 32 bits pois é o tamanho do registrador de instrução.

b) Como este sistema possui 15 registradores, necessita-se no mínimo de 4 bits para se identificar o registrador. Para endereçar 16M células, são necessários 24 bits. Logo o campo da instrução que identifica o registrador possui 4 bits, o que identifica o endereço de memória 24 bits e sobram 4 bits para o código de operação.

c) Como existem 4 bits para o código de operação, podem existir no máximo 16 códigos de operações diferentes.

5.

a)
```
        add 0 0 1           inicializa K com o valor 0
        addi 0 4 10         inicializa reg. 4 com valor 10 para controlar
                            execução do laço
LOOP    beq 1 4 EXIT        finaliza laço quando K igual a 10
        add 2 1 6           obtém endereço de A[K]
        lw 6 6 0            coloca conteúdo de A[K] no registrador 6
        beq 6 0 IF          verifica se A[K] é igual a 0
        addi 0 6 0          se A[K] é diferente de 0, coloca 0 no
                            reg. que vai ter seu conteúdo carregado em B[K]
        beq 0 0 CARB        vai para instrução de carregamento de B[K]
IF      addi 0 6 1          se A[K] é igual a 0, coloca 1 no
                            reg. que vai ter seu conteúdo carregado em B[K]
CARB    add 3 1 7           obtém endereço da B[K]
        sw 7 6 0            coloca valor em B[K]
        addi 1 1 1          incrementa valor de K
        beq 0 0 LOOP        volta para o laço
EXIT    halt                fim do procedimento
```

6.

a) Devemos transformar a representação para a base 2, para podermos identificar os campos de cada instrução:

```
End     Conteúdo
10      0000000|000|000|000|0000000000000|001
                op  regA regB              destreg      add 0 0 1
11      0000000|001|000|010|0000000000000100
                op  regA regB     desl.                 addi 0 2 4
12      0000000|100|001|010|0000000000000111
                op  regA regB     desl.                 beq 1 2 7
13      0000000|000|011|001|0000000000000|100
                op  regA regB              destreg      add 3 1 4
14      0000000|010|100|101|0000000000000000
```

```
                    op  regA regB      desl.                  lw 4 5 0
15        0000000|000|101|101|0000000000000|101
                    op  regA regB               destreg       add 5 5 5
16        0000000|000|101|101|0000000000000|101
                    op  regA regB               destreg       add 5 5 5
17        0000000|011|100|101|0000000000000000
                    op  regA regB     desl.                   sw 4 5 0
18        0000000|001|001|001|0000000000000001
                    op  regA regB     desl.                   addi 1 1 1
19        0000000|100|000|000|1111111111111000
                    op  regA regB      desl.                  beq 0 0 -8
1A        0000000|110|000|000|0000000000000000
                    op  regA regB      desl.                  halt
```

b)

Registradores:

0-00000000
1-00000004
2-00000004
3-00000020
4-00000023
5-0000000C
6-AC012345
7-12345678

Memória :

End.	Conteúdo	End.	Conteúdo
10	00000001	1B	F0000000
11	00420004	1C	00000000
12	010A0007	1D	00010000
13	00190004	1E	10000000
14	00A50000	1F	7FFF0000
15	002D0005	20	00000008
16	002D0005	21	FFFFFFFC
17	00E50000	22	0000003C
18	00490001	23	0000000C
19	0100FFF8	24	00000010
1A	01800000	25	000000AF