P3 - ORG

1) a) $I(mi + mdLS)$

$$\frac{mi + mdLS}{1 + LS}$$

b) $\frac{(I).\ mi.\ p}{h + mi.\ p}$

2) 16-way
bloco: 64 bytes
$1024 . 2^{10}$ bytes

bloco { +1 : validade, +1 : reference, +1 : dirty }

{ endereço = 48 bits

a) index?

→ 16-way = $\frac{16 \text{ blocos}}{\text{conjunto}}$

→ blocos = $\frac{2^{20} \text{ bytes}}{\text{bloco}} \Big/ \frac{2^6 \text{ bytes}}{\text{bloco}} = 2^{14}$ blocos

→ Size = 2^{20} bytes

→ $\frac{64 \text{ bytes}}{\text{bloco}}$

→ conjuntos = $\frac{2^{14} \text{ blocos}}{2^4 \frac{\text{blocos}}{\text{conjunto}}} = 2^{10}$ conjuntos

$\downarrow \log_2(2^{10})$

10 bits = índice

b) tag?

tag	index	Word offset	Byte offset
	10	4	2 → sempre

= 48 − 10 − 4 − 2

= 32 bits

$\frac{64 \text{bytes}}{\text{bloco}} = \frac{2^6 \text{ WORDS}}{2^2 \frac{\text{WORDS}}{\text{bloco}}} = \frac{16 \text{ WORDS}}{\text{bloco}}$ ($\log_2 16 = 4$)

→ WORD = 4 bytes

c) tamanho em bits de um bloco?

V	R	D	data
1	1	1	64 bytes

Bloco = 1 + 1 + 1 + 64.8 = 547 bits

Cache? Conjunto de: 16 blocos

Conjuntos = $2^4 . 547$ bits = 8752 bits

Cache: 2^{10} conjuntos

Cache = $2^{10} . 8752$ bits = 8752 K bits

d)
- penalidade$_{main}$ = 100 ciclos (leitura)
- penalidade$_{main}$ = 80 ciclos (escrita)
- I-C e I-D, main
- $I = 10^6$
- % store = 10% = 10^{-1}
- write-through + write-buffer → s/ overflow → penalidade$_{buffer}$ = 0 ciclos
- 4 palavras / bloco → $\frac{4 \text{ escritas}}{\text{write miss}} + \frac{4 \text{ leituras}}{\text{write miss}}$
- write allocate
- miss rate$_{I-C}$: 10% = 10^{-1}
- stalls escrita ??
- $\frac{\text{total de escritas}}{\text{programa}} = I . \%S = 10^6 . 10^{-1} = 10^5$

stalls escrita $= 10^5 . 10^{-1} . 8 \ 10 . 4 = 32 \ 10^5$ ciclos

$stalls = 10^5 \cdot 10^{-1} (4 \cdot \cancel{10} + 4 \cdot 100)$

(mr; 10^5 → str prog; 4 escritas; 4 leituras)

→ write miss: tag ≠

↳ substituir e carregar os adjacentes no bloco

↳ buffer: retira a penalidade (s/ overflow!) de escrita na MP (*)

↳ allocate: coloca a penalidade de leitura (MP) do bloco antes de substituir

↳ write-throught + buffer

$\frac{1 \text{ escrita}}{\text{cache}} \rightarrow \frac{1 \text{ escrita}}{MP}$ → buffer (*)

$stalls = 10^5 \cdot 10^{-1} \cdot 4 \cdot 10^2$

$stalls_{escrita} = 4 \cdot 10^6$ ciclos

3) a)

```
Sum: add $v0, $0, $0
loop: lw   $t0, 0($a0)
      add  $v0, $t0, $v0
      addi $a0, $a0, -4
      bne  $a0, $0, loop
      jr $ra
```

loop unrolling

Ⓘ

```
lw   $t0, 0($a0)
add  $v0, $t0, $v0
addi $a0, $a0, -4
~~bne $a0, $0, loop~~
lw   $t0, 0($a0)
add  $v0, $t0, $v0
addi $a0, $a0, -4
bne  $a0, $0, loop
```

Ⓘⓘ

```
lw   $t0, 0($a0)   ⓐ   → [0]
add  $v0, $t0, $v0 ⓑ
addi $a0, $a0, -8
lw   $t0, 0($a0)       → [-4]
add  $v0, $t0, $v0
bne  $a0, $0, loop
```

Full-Forwarding

ⓐ IF ID EX ME WB ($t0)

ⓑ IF ID EX ME WB

→ olhar arquivo 3_a.ods!

Ⓘⓘⓘ

```
lw   $t0, 0($a0)
addi $a0, $a0, -8
add  $v0, $t0, $v0
lw   $t1, 4($a0)
add  $v0, $t1, $v0
bne  $a0, $0, loop
```

b) 8 instruções (→ 4 bytes)

$8 \times 4 = 32$ bytes

c) Emissão dinâmica: os nops desaparecem, pois o HW vai emitir quando puder e irá parar quando necessário.

Logo: 6 instruções. $6 \times 4 = 24$ bytes

d) $\frac{2 \cdot 10^8 \text{ iterações (original)}}{2} = 10^8$ iterações c/ loop unrolling

↳ $\frac{6 \text{ instruções úteis}}{\text{iteração}} \Rightarrow 6 \cdot 10^8$ instruções

e) 6 instruções e 4 ciclos

$CPI_{ideal\,(cache)} = 4/6 = 2/3$

$CPI = \frac{\text{ciclos}}{\text{instrução}}$

↳ $0,\bar{6} < CPI_{ideal\,(cache + hazardo)} = 0,5$ (→ 2-issue)

↳ faz sentido.

f) $mr = 0,01 = 10^{-2}$ $\quad \%LS = \frac{1}{4} = 0,25$

$p = 100$ ciclos

$CPI_{total} = CPI_{ideal} + CPI_{stall}$ → $\frac{ciclos_{stall}}{I}$

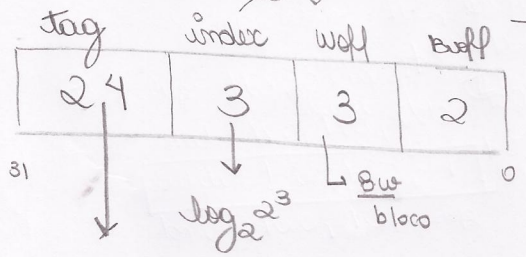
$CPI_{total} = \frac{2}{3} + \frac{1}{4} = \frac{2^3 4 + 1 3}{12}$

$ciclos_{stall} = (I \cdot \frac{1}{4}) \cdot 10^{-2} \cdot 10^2 = \frac{I}{4}$ "

$CPI_{total} = \frac{11}{12}$ "

4) • 2-way • 128 palavras • $\frac{8 \text{ palavras}}{\text{bloco}}$ • LRU

↳ $\frac{2 \text{ blocos}}{\text{conjunto}}$ " → $\frac{128}{8} = \frac{2^7}{2^3} = 2^4$ blocos



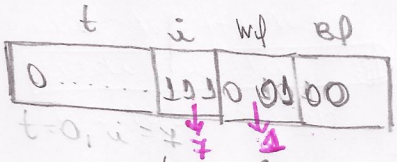
2^4 blocos = 2^3 conjuntos / 2^1

$32 - 3 - 3 - 2 = 24$

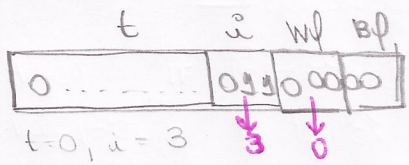
Endereços = (I) 0xEA, (II) 0x60, (III) 0x1FC, (IV) 0x3E8, (V) 0xFC

↳ byte address!

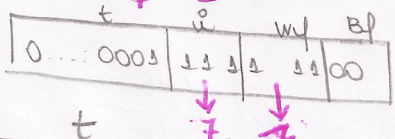
(I) 0xEA:



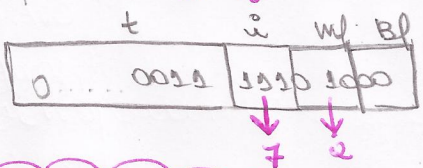
(II) 0x60:



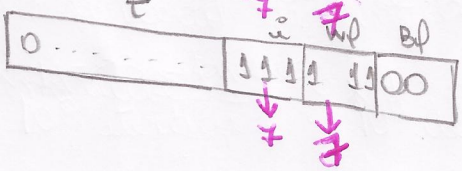
(III) 0x1FC:



(IV) 0x3E8:



(V) 0xFC:



• olhar o arquivo 4.ods!

5)

```
    addi $s0, $0, 0
    addi $s1, $0, 8    (?)
loop: ...
    addi $s0, $s0, 1   # $s0++
    bne $s1, $0, loop  # se $s1 ≠ 0 → loop
    addi $s1, -4       #
```