

# Exame 2022

$$\textcircled{1} \quad 6288_{10}, 5430_{10}$$

$$\begin{array}{r}
 6288 \xrightarrow{16} 393 \\
 -48 \\
 \hline
 140 \\
 -144 \\
 \hline
 48 \\
 -48 \\
 \hline
 \end{array}
 \quad
 \begin{array}{r}
 393 \xrightarrow{16} 24 \\
 -32 \\
 \hline
 73 \\
 -64 \\
 \hline
 9
 \end{array}
 \quad
 \begin{array}{r}
 24 \xrightarrow{16} 1 \\
 -16 \\
 \hline
 8
 \end{array}$$

$6288_{10} = 0000100001110000_2$

$$16 \times 3 = 48 \quad 16 \times 5 = 80$$

$$16 \times 9 = 144$$

$$16 \times 3 = 48$$

$$16 \times 4 = 64$$

$$5430_{10} = \boxed{0000110001100000_2}$$

$$\begin{array}{r}
 5430 \xrightarrow{16} 339 \\
 -48 \\
 \hline
 63 \\
 -48 \\
 \hline
 150 \\
 -144 \\
 \hline
 6
 \end{array}
 \quad
 \begin{array}{r}
 339 \xrightarrow{16} 21 \\
 -32 \\
 \hline
 19 \\
 -16 \\
 \hline
 3
 \end{array}
 \quad
 \begin{array}{r}
 21 \xrightarrow{16} 1 \\
 -16 \\
 \hline
 1
 \end{array}
 \quad
 \begin{array}{r}
 1 \xrightarrow{16} 0 \\
 -0 \\
 \hline
 0
 \end{array}$$

R:  $6288_{10}, 0000110001100000_2$   
 os últimos 2 bits  
 estão a zero //

② Michael Flynn classificou os processadores em 2 categorias: instrução e dados. Ele divide em SISD, SIMD, MIMD e MISD. SISD é single instruction e single data, ou seja, o processador só faz uma instrução para um bloco de dados. SIMD é single instruction multiple data, executa a mesma inst. para vários dados, normalmente usado em processadores vetoriais e GPUs. MIMD Multiple Instruction Multiple Data executa várias instruções em vários dados normalmente usado em processadores multicore. MISD multiple instruction single data executa várias instruções num só set de dados, algo que não existe comercialmente. O processador Single core é SISD porque só realiza uma instrução num set de dados de cada vez.

③ Pipeling consiste na divisão de uma supertask em sub-tasks menores, independentes e que demoram o mesmo tempo; com isto podemos lancer novas instruções a cada ciclo de relógio. No entanto uma instrução também é finalizada a todo ciclo do relógio. Desta modo, conseguimos executar n instruções em  $K(n+k) + (n-1)$  em vez de  $nK$  (non-pipeline). Apesar disto o pipelining não muda o tempo de execução de uma instrução porque o

$$\sum T_{sub} > T_{super\ task}$$

④ Um hazard estrutural acontece quando duas unidades de hardware são requeridas ao mesmo tempo. Uma unidade single integer nunca é usada duas vezes no mesmo tempo porque só é requerida no estado Exec e só demora 1 ciclo de relógio. Temos múltiplas unidades FP, que podem demorar mais de 1 ciclo de relógio, assumimos que temos unidades suficientes para executar todos

as instruções de FP requeridas, caso contrário, necessitaremos de criar stalls entre instruções. O recurso que pode ser requerido duas vezes ao mesmo tempo é a memória, no estado MEM e estado IF para dar fetch de dados e instruções. A solução para este problema seria ter 2 memórias, uma para instruções e outra para dados (arquitetura Harvard).

→ Um hazard impede que a próxima instrução seja executada a tempo

*Referir no inicio*

⑤ Num pipeline de 5 estados clássico, as únicas hazards que não acontecem são WAW e WAR. Isto acontece porque o pipeline clássico não tem instruções longas de





