# TAREFAS

**TAREFA:** Replique a simulação da Figura 3 no seu computador. Para o fazer, siga os passos seguintes (conforme descrito em pormenor na Secção 7 do GSG):

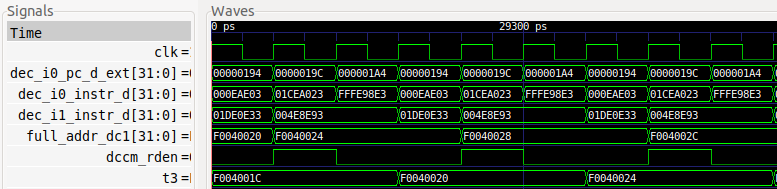
* Se necessário, gere o binário de simulação (*Vrvfpgasim*).
* No PlatformIO, abra o projeto fornecido em: *[RVfpgaPath]/RVfpga/Labs/Lab19/LW-SW\_Instruction\_ExtMemory*.
* Defina o caminho correto para o binário de simulação RVfpga (*Vrvfpgasim*) no ficheiro *platformio.ini*.
* Gere o trace da simulação utilizando o Verilator (Generate Trace).
* Abra o trace no GTKWave.
* Use o ficheiro *test\_Blocking\_Extended.tcl* (fornecido em *[RVfpgaPath]/RVfpga/Labs/Lab19/LW-SW\_Instruction\_ExtMemory*) para abrir os mesmos sinais que os mostrados na Figura 6. Para o efeito, no GTKWave, clique em *File → Read Tcl Script File* e selecione o ficheiro *test\_Blocking\_Extended.tcl*.
* Clique em *Zoom In* () várias vezes e analise a região que começa em 42500 ps.

Solução apresentada no documento principal do Lab 19.

**TAREFA:** Usando os Contadores HW, meça o número de ciclos, instruções, leituras e escritas no programa da Figura 2. Quanto tempo no total (tanto para leitura como para escrita) demora a aceder à memória externa DDR? Pode comparar a execução quando utiliza a memória DDR como na Figura 3 e quando utiliza a DCCM (outro projeto PlatformIO é fornecido em *[RVfpgaPath]/RVfpga/Labs/Lab19/LW-SW\_Instruction\_DCCM/*, que contém o mesmo programa preparado para ler/escrever na DCCM). Lembre-se de que a memória simulada não é a mesma que a memória DDR real na placa Nexys A7.

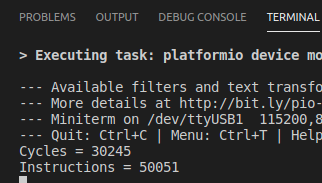
**DCCM**:

Simulação no Verilator:



Cada iteração executa 5 instruções em 3 ciclos. Só se perde meio ciclo por iteração.

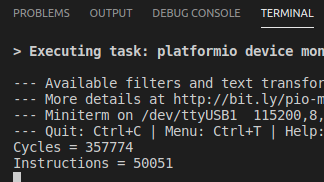
Execução na placa:



Ciclos por iteração = 3

**DDR Memory**:

Execução na placa:



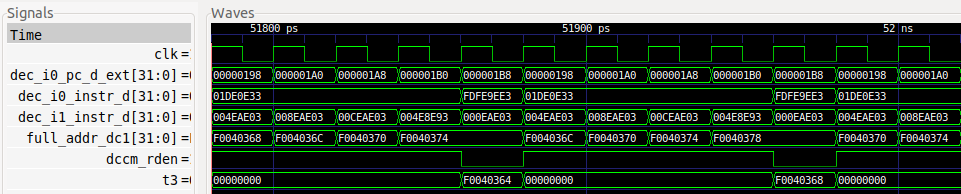
O número de instruções é o mesmo, uma vez que o programa é o mesmo. No entanto, agora são necessários cerca de 358000 ciclos para executar todas as iterações, portanto:

Número de ciclos gastos a aceder à memória por iteração ≈ (358000 - 30000) / 10000 ≈ 33

**TAREFA:** Use o exemplo de *[RVfpgaPath]/RVfpga/Labs/Lab19/LW\_Instruction\_ExtMem* para estimar a latência de leitura da memória externa DDR usando os contadores HW. Tal como na tarefa anterior, pode utilizar o exemplo de *[RVfpgaPath]/RVfpga/Labs/Lab19/LW\_Instruction\_DCCM* para comparar com um programa sem paragens devido aos acessos à memória. Lembre-se de que a memória simulada não é a mesma que a memória DDR real na placa Nexys A7.

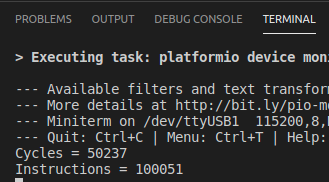
**DCCM**:

Simulação no Verilator:



Cada iteração executa 10 instruções em 5 ciclos, pelo que executa com o IPC ideal.

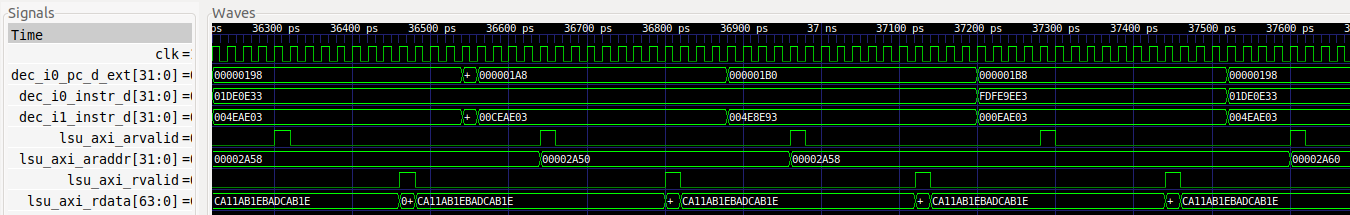
Execução na placa:



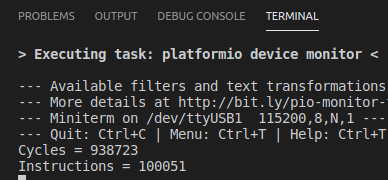
Ciclos por iteração = 5

**Memória DDR**:

Simulação no Verilator:



Execução na placa:

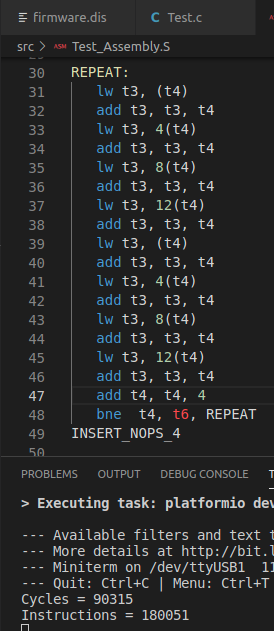


O número de instruções é o mesmo, uma vez que o programa é o mesmo. No entanto, agora são necessários cerca de 939000 ciclos para executar todas as iterações, portanto:

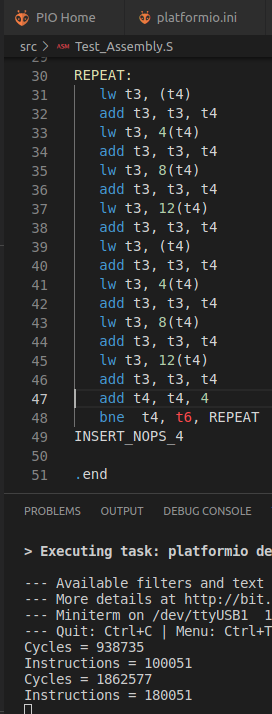
Latência de uma leitura da memória DDR ≈ (939000 - 50000) / (10000 \* 4) ≈ 22

Para verificar se está correto, duplicamos o número de instruções de leitura e executamos o programa novamente:

**DCCM**:



**Memória DDR**:



Latência de uma leitura da memória DDR ≈ (1862000 - 90000) / (10000 \* 8) ≈ 22

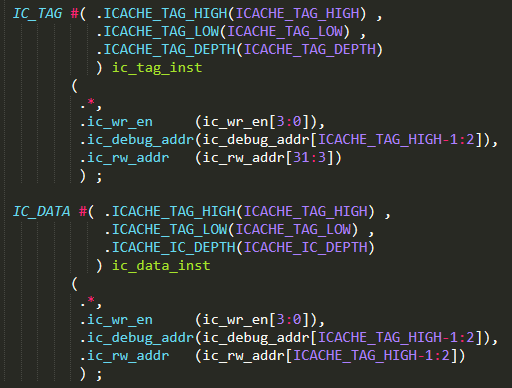
**TAREFA:** Um exercício bastante complexo mas muito interessante é analisar o Controlador de Memória utilizado no Sistema RVfpga. Lembre-se que pode encontrar os módulos que compõem este controlador na pasta *[RVfpgaPath]/RVfpga/src/LiteDRAM*, e que o módulo de topo está implementado no ficheiro *litedram\_top.v* dentro dessa pasta. Pode começar com a simulação da Figura 3 e adicionar e analisar alguns sinais do controlador da LiteDRAM.

Solução não fornecida.

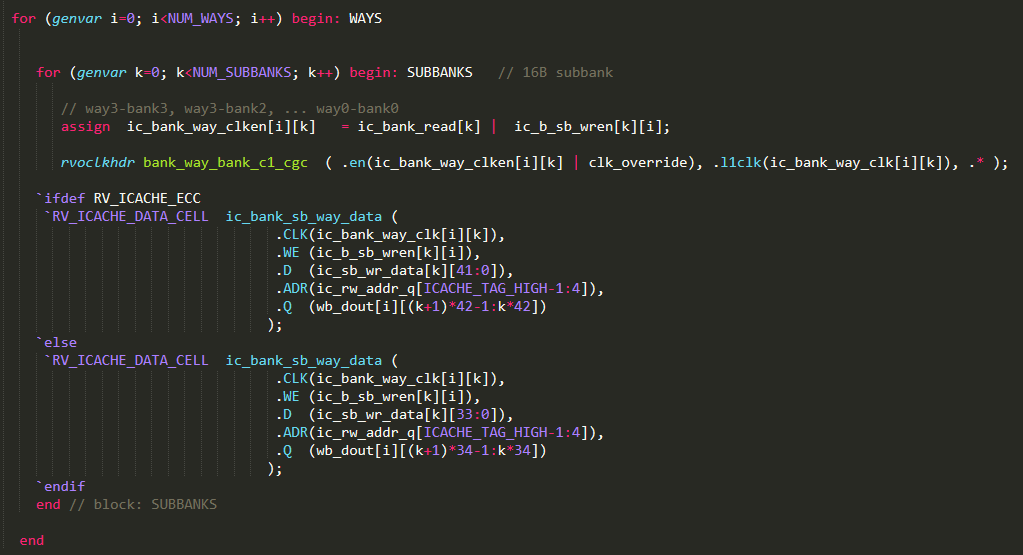
**TAREFA:** Analise o módulo **ifu\_ic\_mem** para compreender como os elementos da Figura 4 estão implementados.

Módulo **ifu\_ic\_mem**:

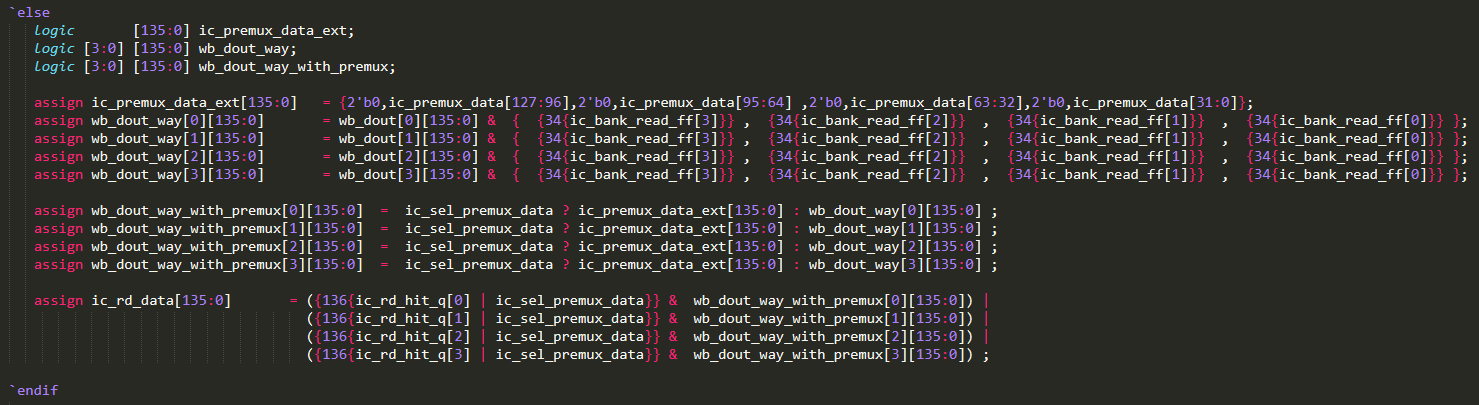
Instanciação do Data Array e do Tag Array:



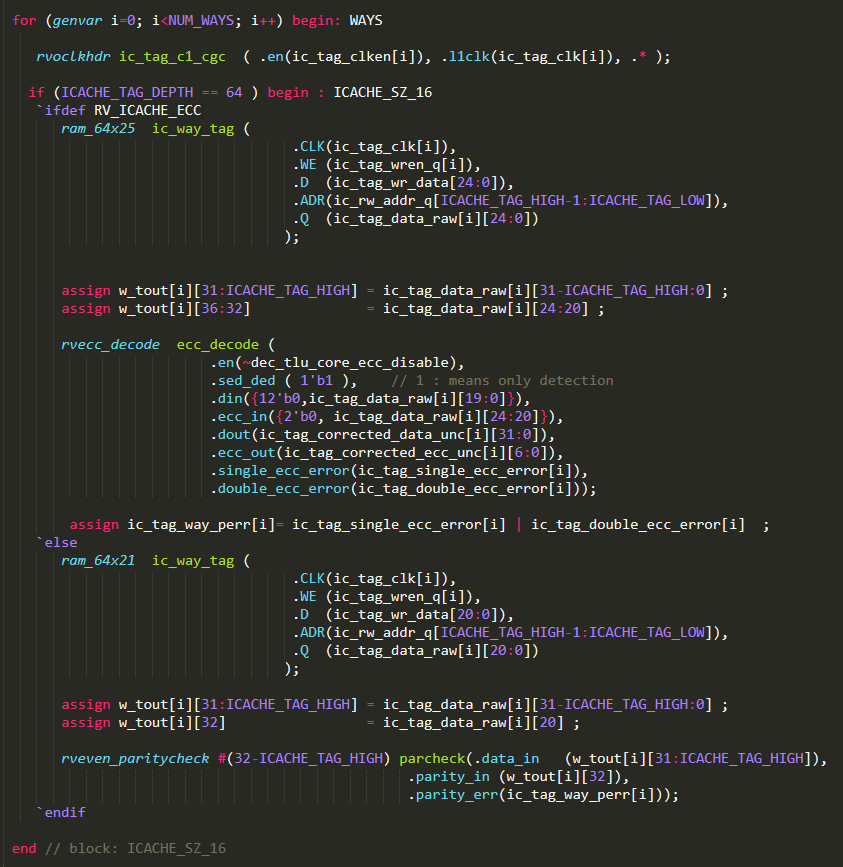
Data Array e bits Parity (No nosso caso, o RV\_ICACHE\_ECC não está definido):



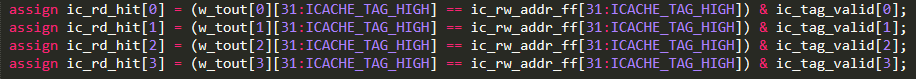
Multiplexer 4-1:



Tag Array e bits Parity (No nosso caso, RV\_ICACHE\_ECC não está definido):



Comparadores:



**TAREFA:** Replique a simulação da Figura 6 no seu computador. Para o fazer, siga os passos seguintes (descritos em pormenor na Secção 7 do GSG):

* Se necessário, gere o binário de simulação (Vrvfpgasim).
* No PlatformIO, abra o projeto fornecido em: *[RVfpgaPath]/RVfpga/Labs/Lab19/InstructionMemory\_Example*.
* Atualize o caminho para o binário de simulação RVfpga (*Vrvfpgasim*) no ficheiro *platformio.ini*.
* Gere o trace da simulação com o Verilator (Generate Trace).
* Abra o trace no GTKWave.
* Use o ficheiro *test1\_Miss.tcl* (fornecido em *[RVfpgaPath]/RVfpga/Labs/Lab19/InstructionMemory\_Example*) para abrir os mesmos sinais que os mostrados na Figura 6. Para isso, no GTKWave, clique em File → Read Tcl Script File e selecione o ficheiro *test1\_Miss.tcl*.
* Clique em *Zoom In* () várias vezes e analise a região de 28900 ps a 30220 ps.

Também pode analisar algumas coisas com mais pormenor, como a escrita na I$ ou o bypass das instruções iniciais.

Solução apresentada no documento principal do Lab 19.

**TAREFA:** Reproduza a simulação da Figura 7 no seu computador. Utilize o ficheiro *test1\_Hit.tcl* (em *[RVfpgaPath]/RVfpga/Labs/Lab19/InstructionMemory\_Example*). *Zoom In* () várias vezes e passe para 34680ps.

Solução apresentada no documento principal do Lab 19.

**TAREFA:** Analise o código Verilog da Figura 9 e explique o seu funcionamento com base nas explicações anteriores.

Solução não fornecida.

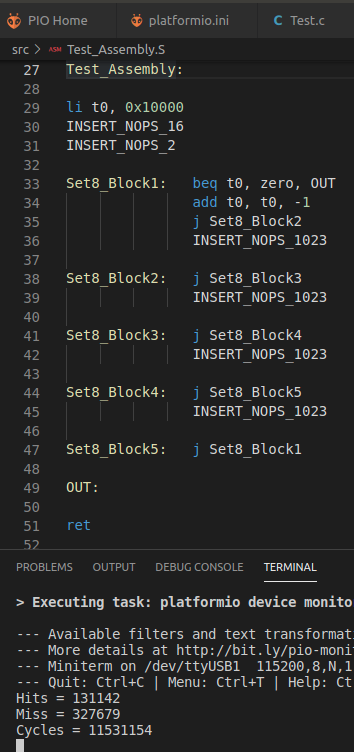
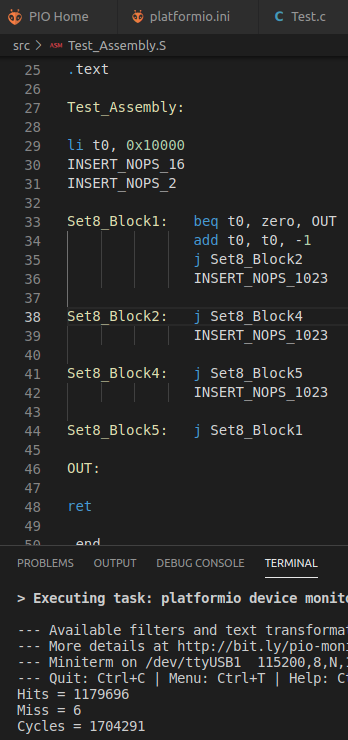
**TAREFA:** Analise o código Verilog da Figura 10 e explique o seu funcionamento com base nas explicações anteriores.

Solução não fornecida.

# EXERCÍCIOS

1. Transforme o ciclo infinito da Figura 11 num ciclo com 0x10000 iterações, mas mantenha as instruções j nos mesmos endereços. Meça o número de ciclos e I$ hits e misses. Em seguida, remova uma das instruções j e meça as mesmas métricas. Compare e explique os resultados.

**5 instruções jump: 4 instruções jump:**

No programa com 4 instruções j, o número de I$ misses e o número de ciclos diminuem drasticamente, pois agora os blocos não entram em conflito uns com os outros. Ao mesmo tempo, o número de I$ hits aumenta bastante.

1. Utilize o programa da Figura 5 para analisar um acerto I$ do ponto de vista da Política de Substituição I$.

Solução não fornecida.

1. Expanda a Figura 6 para analisar em pormenor a forma como cada pedaço de 64 bits é escrito na I$.

Solução não fornecida.

1. Analise em simulação e na placa outras configurações da I$, como uma I$ com um tamanho de bloco diferente. Lembre-se que o número de vias não pode ser modificado.

Solução não fornecida.

1. Analise a lógica que verifica a correção da informação de paridade do Data Array e do Tag Array.

Solução não fornecida.