# TAREFAS

**TAREFA:** Examine os elementos do processador incluídos na Figura 1 no código Verilog e explique o seu funcionamento.

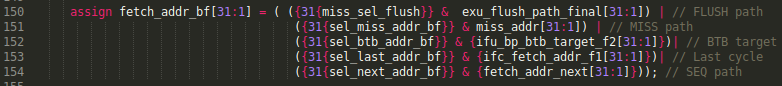
* Os elementos apresentados no andar Decode (Register File, Registo de Instruções e Unidade de Controlo) podem ser encontrados nos módulos **dec**, **dec\_decode\_ctl** e **dec\_gpr\_ctl**.
* Os elementos apresentados no andar EX1 podem ser encontrados nos módulos **exu** e **exu\_alu\_ctl**.
* Os elementos apresentados no andar FC1 podem ser encontrados nos módulos **ifu** e **ifu\_ifc\_ctl**.

**Andar FC1:**

* Multiplexer 2:1: Módulo **ifu\_ifc\_ctl**



* Multiplexer 5:1: Módulo **ifu\_ifc\_ctl**



* Somador para endereços sequenciais: Módulo **ifu\_ifc\_ctl**



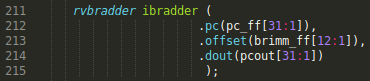
**EX1 Stage:**

* Comparador: Módulo **exu\_alu\_ctl**



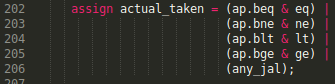
Compara os dois operandos:

* Se são iguais: eq=1.
* Se são diferentes: eq=0.
* Somador para o endereço de destino do salto: Módulo **exu\_alu\_ctl**



Calcula a soma do PC com o offset.

* LOGICA: Módulo **exu\_alu\_ctl**



actual\_taken contém a resolução da direção do salto: 1 se o salto tiver de ser efectuado e 0 se não tiver de ser efectuado. Por exemplo:

* + Se a instrução for um beq (ap.beq==1) e os dois operandos forem iguais (eq==1) 🡪 actual\_taken = 1
  + Se a instrução for um bne (ap.bne==1) e os dois operandos forem diferentes (ne==1) 🡪 actual\_taken = 1
  + Se a instrução for um jal (any\_jal==1) o salto tem de ser efetuado 🡪 actual\_taken = 1



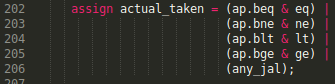
O salto foi mal previsto (cond\_mispredict=1) se foi previsto que fosse tomado (ap.predict\_t = 1) e não deve ser tomado (actual\_taken = 0), ou se foi previsto que não deve ser tomado (ap.predict\_nt = 1) e deve ser tomado (actual\_taken = 1)



O pipeline deve ser descarregado se tiver sido previsto incorretamente (cond\_mispredict=1), a instrução é válida (valid\_ff=1), e o pipeline não está a ser descarregado ou bloqueado.

**TAREFA:** Explique como o sinal flush\_upper é gerado no módulo **exu\_alu\_ctl** a partir do sinal eq, sinais de controlo ap.beq, ap.predict\_t e ap.predict\_nt, e outros sinais.

* LOGICA: Módulo **exu\_alu\_ctl**



actual\_taken contém a resolução da direção do salto: 1 se o salto tiver de ser efectuado e 0 se não tiver de ser efectuado. Por exemplo:

* + Se a instrução for um beq e os dois operandos forem iguais 🡪 actual\_taken = 1
  + Se a instrução for um bne e os dois operandos forem diferentes 🡪 actual\_taken = 1
  + Se a instrução for um jal o salto tem de ser efetuado 🡪 actual\_taken = 1



O salto foi mal previsto (cond\_mispredict=1) se foi previsto que fosse tomado (ap.predict\_t = 1) e não deve ser tomado (actual\_taken = 0), ou se foi previsto que não deve ser tomado (ap.predict\_nt = 1) e deve ser tomado (actual\_taken = 1)



O pipeline deve ser descarregado se tiver sido previsto incorretamente (cond\_mispredict=1), a instrução é válida (valid\_ff=1), e o pipeline não está a ser descarregado ou bloqueado.

**TAREFA:** Analise no código Verilog o efeito dos sinais exu\_flush\_final, exu\_flush\_upper\_e2, exu\_i0\_flush\_final e exu\_i1\_flush\_final em EX1 e nos andares que o precedem: FC1, FC2, Align e Decode. Para esta análise, pode ser útil utilizar as simulações da Secção 2.B, onde pode incluir os sinais de que necessita.

Solução não fornecida.

**TAREFA:** Modifique a figura 1 para incluir os valores de cada sinal indicado na figura 3 nos ciclos *i*, *i+1*, e *i+2*.



**TAREFA:** Modifique o programa da Figura 2 para fazer com que a primeira instrução de salto obtenha os seus operandos de entrada através de forwarding.

Solução não fornecida.

**TAREFA:** Modifique a figura 1 para incluir os valores de cada sinal indicado na figura 4 nos ciclos *i*, *i+1*, e *i+2*.



**TAREFA:** Analise o funcionamento dos dois multiplexers da FC1 com o exemplo da Figura 2, examinando os sinais em diferentes circunstâncias.

Por exemplo, analise como é feito o Fetch para uma execução sequencial (ou seja, um grupo de instruções sem saltos). Verá que, no processador SweRV EH1, o funcionamento neste caso é o seguinte:

- Nos ciclos pares, o fetch\_addr\_next é selecionado usando o multiplexer 5:1, que contém o Fetch Address atual (ifc\_fetch\_addr\_f1) mais 16, lendo assim o próximo pacote sequencial de instruções de 128 bits (lembre-se que uma leitura da I$ fornece 128 bits).

- Nos ciclos ímpares, o ifc\_fetch\_addr\_f1 é selecionado utilizando o multiplexer 5:1, pelo que não são carregadas novas instruções.

Desta forma, quatro instruções de 32 bits são carregadas a cada 2 ciclos, que é a mesma taxa de instruções necessárias ao andar Decode (2 instruções por ciclo).

Note que nos processadores da DDCARV o PC é simplesmente incrementado de quatro em cada ciclo (para execução sequencial) para ir buscar uma instrução por ciclo.

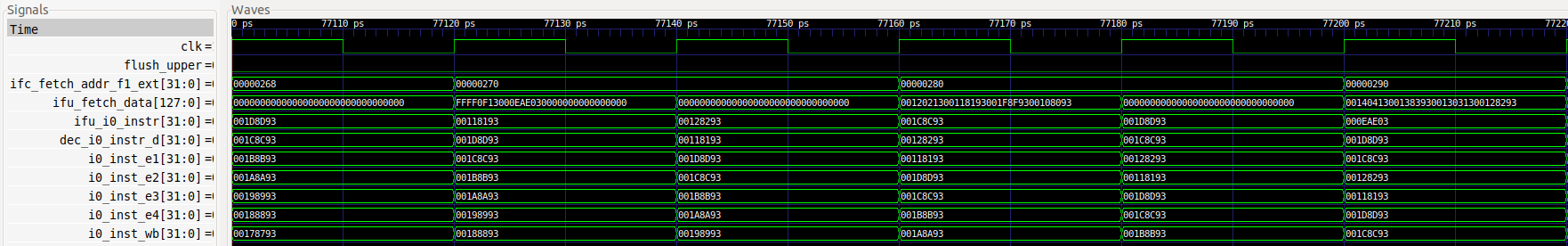
Modifique também o programa da Figura 2 para criar novos cenários. Por exemplo, pode adicionar algumas instruções A-L após o salto tomado e ver como são descarregadas após o redirecionamento.

**EXECUÇÃO SEQUENCIAL:**

Utilize as seguintes fontes:

* Programa de: *[RVfpgaPath]/RVfpga/Labs/Lab14/LW\_Instruction\_ExtMemory*
* Tcl Script de: *[RVfpgaPath]/RVfpga/Labs/RVfpgaLabsSolutions/Programs\_Solutions/Lab16/test\_SequentialExecution.tcl*

Pode obter a seguinte simulação no Verilator:



268: 000eae03 lw t3,0(t4)

26c: ffff0f13 addi t5,t5,-1

270: 00108093 addi ra,ra,1

274: 001f8f93 addi t6,t6,1

278: 00118193 addi gp,gp,1

27c: 00120213 addi tp,tp,1

280: 00128293 addi t0,t0,1

284: 00130313 addi t1,t1,1

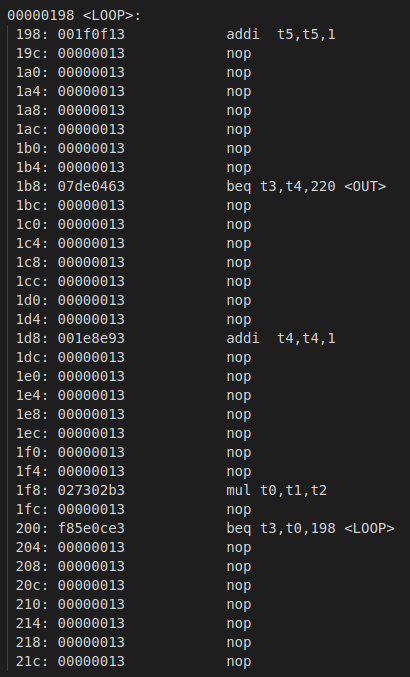
288: 00138393 addi t2,t2,1

28c: 00140413 addi s0,s0,1

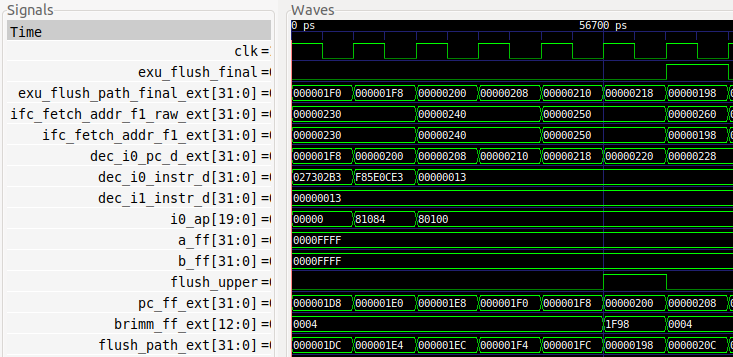
Podemos ver que a cada dois ciclos é carregado um novo pacote de 128 bits.

**TAREFA:** No Lab 15, analisámos como os conflitos de dados RAW são resolvidos no andar Commit através das ALUs secundárias. À semelhança das instruções A-L que estudámos nesse laboratório, uma instrução de salto condicional pode ter um conflito de dados RAW com uma operação multi-ciclo anterior que tem de ser resolvido no momento do Commit. Se for determinado que o salto foi mal previsto, o pipeline deve ser descarregado e redirecionado a partir do andar Commit. Analise esta situação utilizando uma versão ligeiramente modificada do programa da Figura 2, fornecida em *[RVfpgaPath]/RVfpga/Labs/Lab16/BEQ\_Instruction\_HazardCommit*, e o script *.tcl* fornecido nessa mesma pasta.

**Código gerado:**



**Simulação no Verilator:**



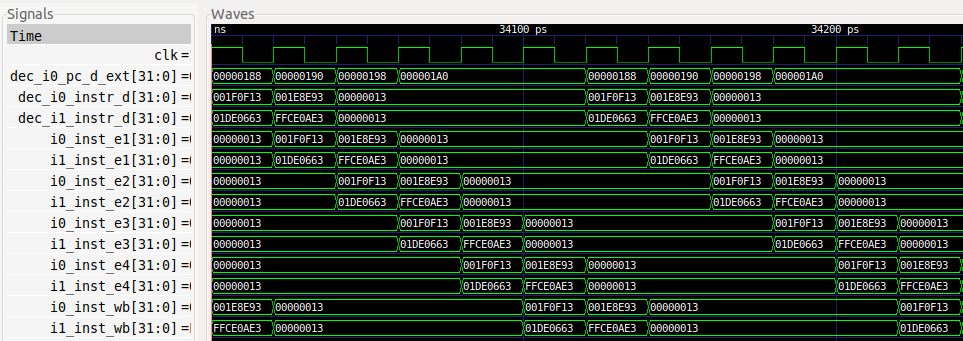
A instrução beq (0xf85e0ce3) é descodificado, passa por EX1 (onde executa nos operandos errados), passa por EX2 e EX3, e depois passa por Commit onde executa novamente nos operandos correctos, desencadeando uma descarga e um redireccionamento (flush\_upper = exu\_flush\_final = 1).

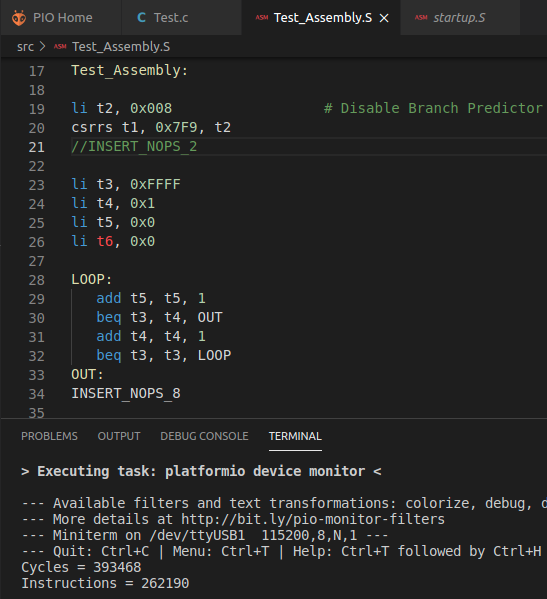
**TAREFA:** No exemplo da Figura 2, remova todas as instruções nop e analise a simulação. Em seguida, calcule o IPC com os Performance Counters, executando o programa na placa.

Active o preditor de saltos utilizado no SweRV EH1 (comentando as duas instruções iniciais da Figura 2) e analise a simulação e a execução na placa.

Compare as duas experiências e explique os resultados.

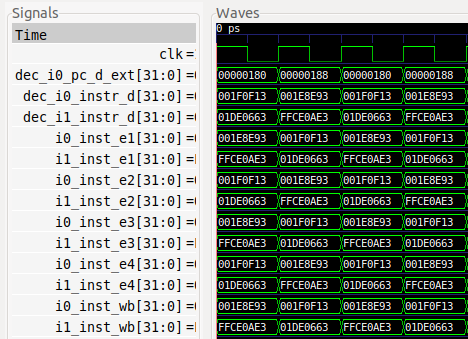
Preditor de saltos Naïve:

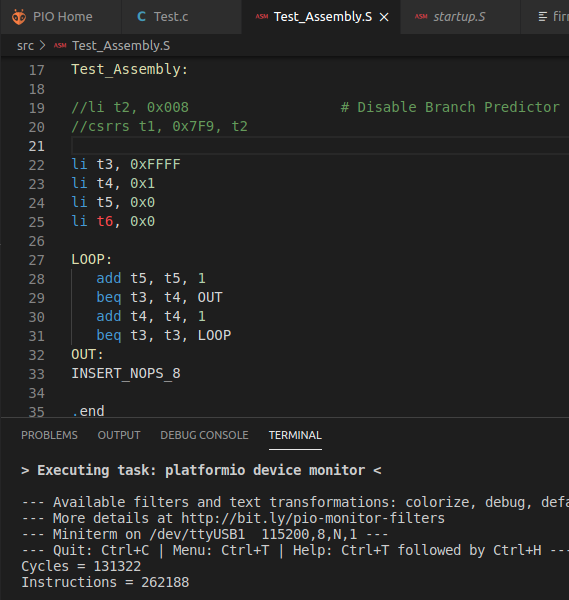




IPC = 262 / 393 = 0.67

Preditor de saltos Gshare:





IPC = 262 / 131 = 2

O IPC é ideal quando utiliza o BP Gshare, mas está longe de ser ideal quando utiliza o BP Naïve devido à descarga e ao redirecionamento causados pela segunda instrução de salto.

**TAREFA**: Analise todos estes módulos de hashing e tente ter uma ideia de como funcionam e como são utilizados nas estruturas BP do Gshare.

Solução não fornecida.

**TAREFA**: Analise como é efetuado o acesso a estas duas estruturas.

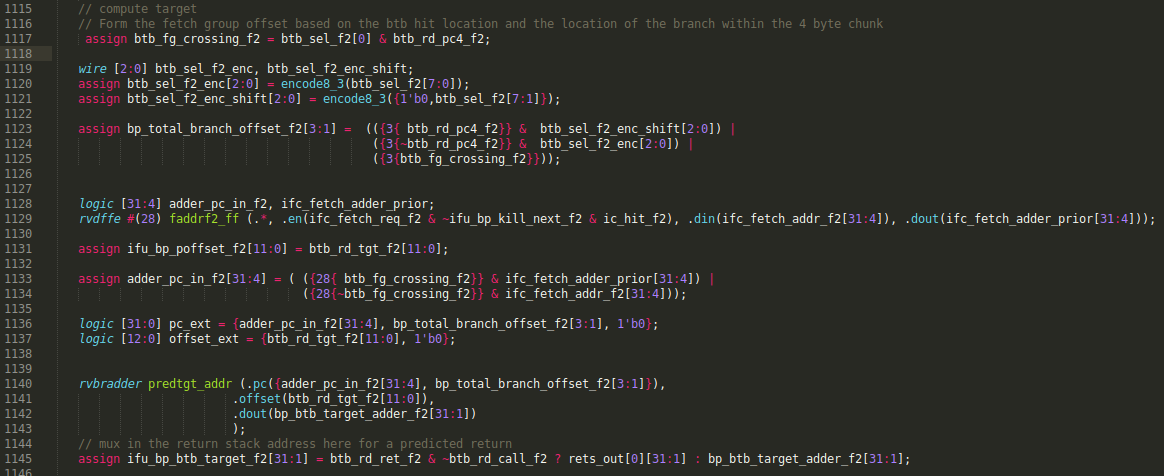
Solução não fornecida.

**TAREFA**: Analise como é calculado o sinal de seleção do multiplexer 5:1.

Solução não fornecida.

**TAREFA**: Analise a forma como o endereço-destino previsto (ifu\_bp\_btb\_target\_f2) é obtido a partir do valor lido na BTB (btb\_rd\_tgt\_f2[11:0]) e o endereço de Fetch em FC2 (ifc\_fetch\_addr\_f2[31:4]).

Módulo **ifu\_bp\_ctl**:



**TAREFA**: Analise o RAS implementado no processador SweRV EH1. Uma pesquisa na Internet fornecerá também informações adicionais sobre o funcionamento desta estrutura (por exemplo <http://www-classes.usc.edu/engr/ee-s/457/EE457_Classnotes/ee457_Branch_Prediction/EE560_05_Ras_Just_FYI.pdf>).

Solução não fornecida.

**TAREFA**: Analisar a forma como o Global History Register é atualizado.

Solução não fornecida.

# EXERCÍCIOS

1. 1) Implemente um preditor de saltos bimodal e compare o seu desempenho em relação ao BP Gshare.

Solução não fornecida.

1. (*O exercício seguinte baseia-se no exercício 4.25 do livro “Computer Organization and Design – RISC-V Edition”, by Patterson & Hennessy ([HePa]).*)

Considere o seguinte ciclo:

LOOP: lw x10, 0(x13) lw x11, 4(x13) add x12, x10, x11 add x13, x13, -8 bnez x12, LOOPAssuma que é utilizada uma previsão perfeita de saltos (no caso do SweRV EH1, podemos emular este comportamento evitando simplesmente a primeira iteração), que o pipeline tem suporte de forwarding completo (mais uma vez, é o caso do SweRV EH1) e que os saltos são resolvidos no andar EX1.

* 1. Apresente uma simulação para a segunda e terceira iterações deste ciclo. Explique o comportamento obtido. Pode utilizar o programa fornecido em *[RVfpgaPath]/RVfpga/Labs/Lab16/HePa\_Exercise-4-25*.

