# TAREFAS

**TAREFA:** Verificar se esses 32 bits (0x0042a303) correspondem à instrução lw t1,4(t0) na arquitetura RISC-V.

**0x0042a303🡪 000000000100 00101 010 00110 0000011**

**imm11:0 = 000000000100**

**rs1 = 00101 = x5 (t0)**

**funct3 = 010**

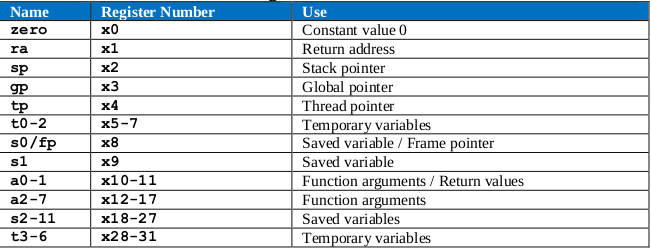
**rd = 00110 = x6 (t1)**

**op = 0000011**

Do Apêndice B do DDCARV:





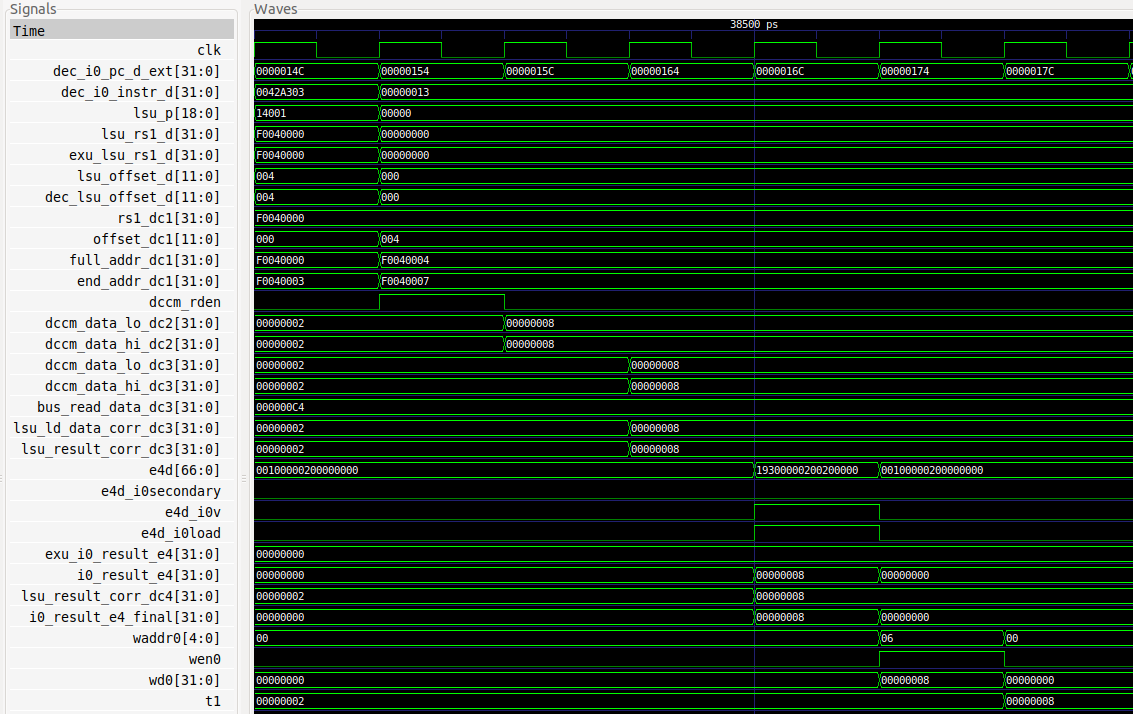


**TAREFA:** Replicar a simulação da Figura 4 no seu computador. Siga as próximas etapas (conforme descrito em detalhes na Seção 7 do GSG):

* Se necessário, gere o binário de simulação (*Vrvfpgasim*).
* No PlatformIO, abra o projeto fornecido em: *[RVfpgaPath]/RVfpga/Labs/Lab13/LW\_Instruction\_DCCM*.
* Corrija o caminho para o binário de simulação do RVfpga (*Vrvfpgasim*) no ficheiro *platformio.ini*.
* Gere o trace da simulação com o Verilator (Generate Trace).
* Abra o trace usando o GTKWave.
* Use o ficheiro *scriptLoad.tcl* (fornecido em *[RVfpgaPath]/RVfpga/Labs/Lab13/LW\_Instruction\_DCCM/*) para abrir os mesmos sinais que os mostrados na Figura 4. Para isso, no GTKWave, clique em *File → Read Tcl Script File* e selecione o ficheiro *scriptLoad.tcl*.
* Clique em *Zoom In* ( ) várias vezes e vá para 18600ps.

Solução fornecida no documento principal do Lab 13.

**TAREFA:** Amplie a simulação da Figura 4 para incluir os sinais mostrados na Figura 6, que são explicados abaixo.



**TAREFA:** Localize as estruturas e os sinais da Figura 6 nos ficheiros Verilog do processador SweRV EH1.

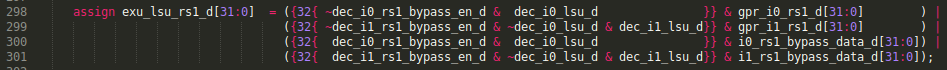
Solução não fornecida.

**TAREFA:** Inclua o sinal *lsu\_p* na simulação da Figura 4 e analise seus bits de acordo com esta descrição.

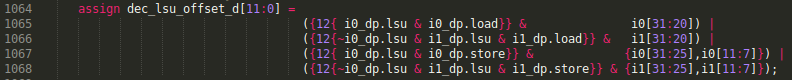
Veja a simulação acima. Podemos ver que, quando a leitura é decodificada, lsu\_p = 0x14001:

* valid = 1. A instrução é válida.
* leitura = 1. É uma leitura.
* palavra = 1. O tamanho do acesso é palavra.

**TAREFA:** Analisar no código Verilog o caminho seguido pelas duas entradas para a LSU (exu\_lsu\_rs1\_d e dec\_lsu\_offset\_d) a partir das fontes onde são obtidas. Vários módulos estão envolvidos nesse processo: **dec**, **exu**, **lsu**. Analise o comportamento desses sinais para obter outras instruções.



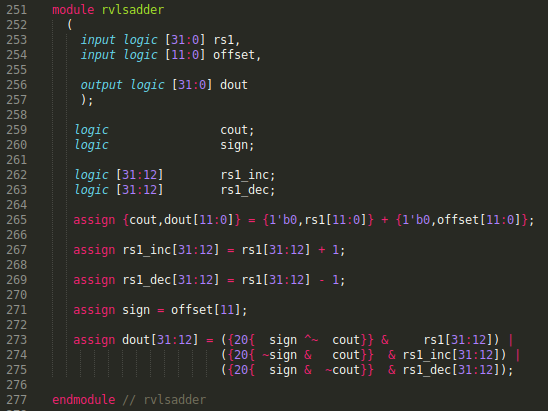
O endereço base pode vir do Register File ou do Bypass, tanto da Via-0 quanto da Via-1.



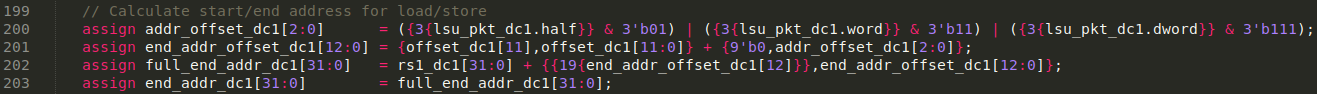
O deslocamento vem dos 32 bits da instrução na Via-0 ou Via-1.

**TAREFA:** Analisar a implementação dos dois somadores do andar DC1, que são instanciados no módulo **lsu\_lsc\_ctl**. Fornecemos orientação na Figura 7 abaixo, mostrando a implementação desses somadores.

**Ficheiro *beh\_lib.sv*:**

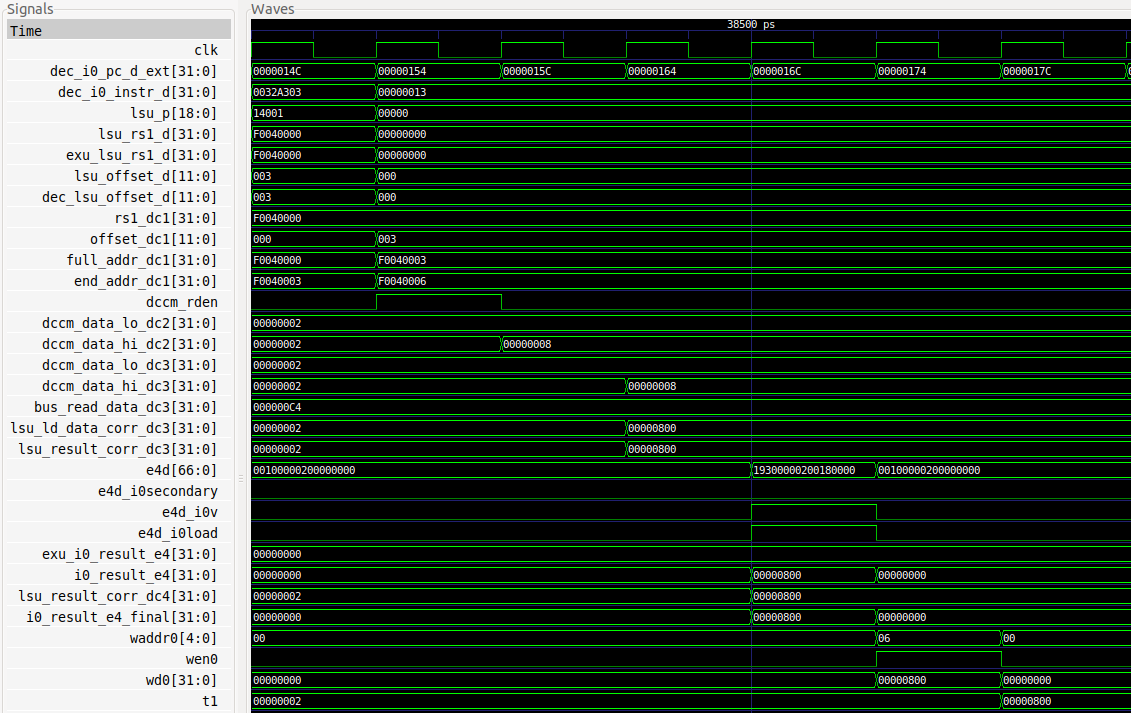


**Ficheiro *lsu\_lsc\_ctl.sv*:**



**TAREFA:** No programa da Figura 2, experimente diferentes tamanhos de acesso (byte, half-word) e acessos não alinhados. Para isso, altere o deslocamento ou o tipo de acesso de lw para lb (*load byte*) ou lh (*load half-word*). Por exemplo, se alterar o deslocamento de 4 para 3, a instrução de leitura de palavra de leitura executará um acesso não alinhado aos 32 bits que começam no endereço 0xF0040003, conforme mostrado na Figura 8. Analise o valor dos sinais lsu\_addr\_dc1[31:0] (ou full\_addr\_dc1[31:0]) e end\_addr\_dc1[31:0] nessas diferentes situações.

No Lab 20, analisamos essa situação a partir dos aspectos internos da DCCM.



Os valores dos sinais lsu\_addr\_dc1[31:0] e end\_addr\_dc1[31:0] comunicam à memória o endereço inicial e final do acesso: 0xF0040003 e 0xF0040007. Duas palavras são lidas (0x00000002 e 0x00000008) e a palavra final é extraída no alinhador (0x00000800).

**TAREFA:** No programa da Figura 2, compare o valor dos sinais dccm\_data\_lo\_dc2[31:0] e dccm\_data\_hi\_dc2[31:0] ao fazer um lw para o endereço 0xF0040004 e para o endereço 0xF0040003.

Acima, pode ver as duas simulações.

* lw para o endereço 0xF0040004

dccm\_data\_lo\_dc2[31:0]: 0x00000008

dccm\_data\_hi\_dc2[31:0]: 0x00000008

Ambos os sinais contêm o valor lido do endereço solicitado.

* lw para o endereço 0xF0040003

dccm\_data\_lo\_dc2[31:0]: 0x00000002 (valor do endereço 0xF0040000)

dccm\_data\_hi\_dc2[31:0]: 0x00000008 (valor do endereço 0xF0040004)

**TAREFA:** Analisar a lógica de Align, Merge, e Error Check usada no código Verilog nos módulos **lsu\_dccm\_ctl** e **lsu\_ecc**.

Solução não fornecida.

**TAREFA:** No programa da Figura 2, compare o valor do sinal lsu\_result\_corr\_dc3[31:0] ao fazer um lw para o endereço 0xF0040004 e para o endereço 0xF0040003.

Acima, pode ver as duas simulações.

* lw para o endereço 0xF0040004

lsu\_result\_corr\_dc3[31:0]: 0x00000008

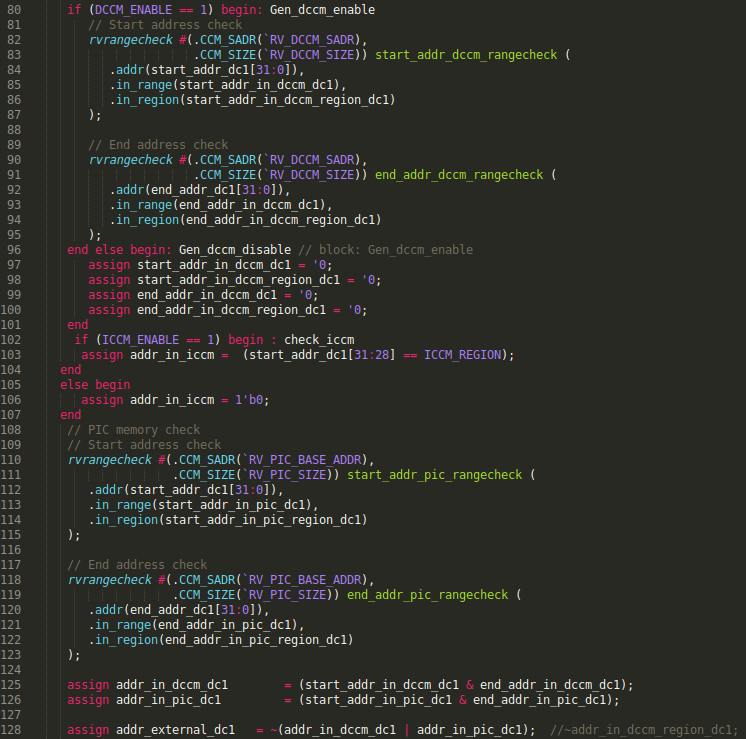
Ele contém o valor lido do endereço solicitado.

* lw para o endereço 0xF0040003

lsu\_result\_corr\_dc3[31:0]: 0x00000800

Ele contém o valor lido do endereço pedido. Tenha em consideração que o RISC-V é little-endian.

**TAREFA:** Analise no código Verilog como o sinal addr\_external\_dc1 foi computado no andar DC1 no módulo **lsu\_addrcheck**.



O módulo **rvrangecheck** é usado para verificar o endereço pedido:

* Se estiver dentro do intervalo de endereços DCCM/ICCM (linhas 80-107), nesse caso o sinal addr\_in\_dccm\_dc1 = 1
* Se estiver dentro do intervalo de endereços do PIC (linhas 108-123), nesse caso o sinal addr\_in\_pic\_dc1 = 1
* Se não estiver em nenhum desses intervalos de endereços, ele estará na memória externa DDR e, nesse caso: addr\_external\_dc1 = 1

**TAREFA:** Verificar se esses 32 bits (0x0062a023) correspondem à instrução sw t1,0(t0) na arquitetura RISC-V.

**0x0062a023🡪 0000000 00110 00101 010 00000 0100011**

**imm11:0 = 000000000000**

**rs2 = 00110 = x6 (t1)**

**rs1 = 00101 = x5 (t0)**

**funct3 = 010**

**op = 0100011**

Do Apêndice B do DDCARV:

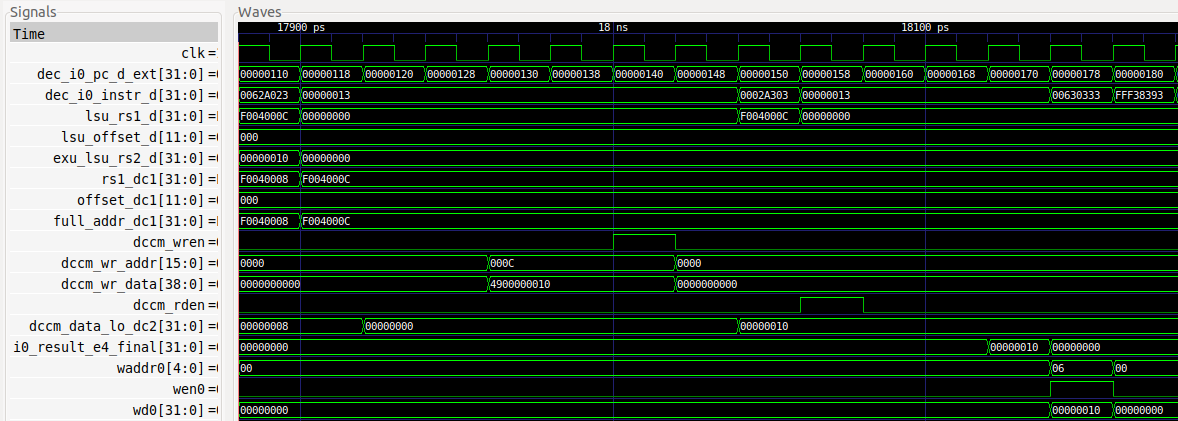


**TAREFA:** Replicar a simulação da Figura 12 no seu computador. Siga as próximas etapas (conforme descrito em detalhes na Seção 7 do GSG):

* Se necessário, gere o binário de simulação (*Vrvfpgasim*).
* Abra no PlatformIO o projeto fornecido em: *[RVfpgaPath]/RVfpga/Labs/Lab13/SW\_Instruction\_DCCM*.
* Atualize o caminho para o binário de simulação do RVfpga (*Vrvfpgasim*) no ficheiro *platformio.ini*.
* Gere o trace da simulação com o Verilator (Generate Trace).
* Abra o trace no GTKWave.
* Use o ficheiro *scriptStore.tcl* (fornecido em *[RVfpgaPath]/RVfpga/Labs/Lab13/SW\_Instruction\_DCCM/*) para exibir os mesmos sinais que os mostrados na Figura 4. Para isso, no GTKWave, clique em *File → Read Tcl Script File* e selecione o ficheiro *scriptStore.tcl*.
* Clique em *Zoom In* ( ) várias vezes e vá para 17900ps.

Solução fornecida no documento principal do Lab 13.

**TAREFA:** Analise na simulação a instrução de leitura que segue a escrita para verificar se o valor foi escrito corretamente na DCCM. precisará adicionar alguns dos sinais da Figura 4 e da Figura 6 para analisar a leitura.



**TAREFA:** Estenda a análise básica realizada nesta seção para a instrução sw por uma via semelhante à análise avançada realizada para a instrução lw na Seção 2.B.

Solução não fornecida.

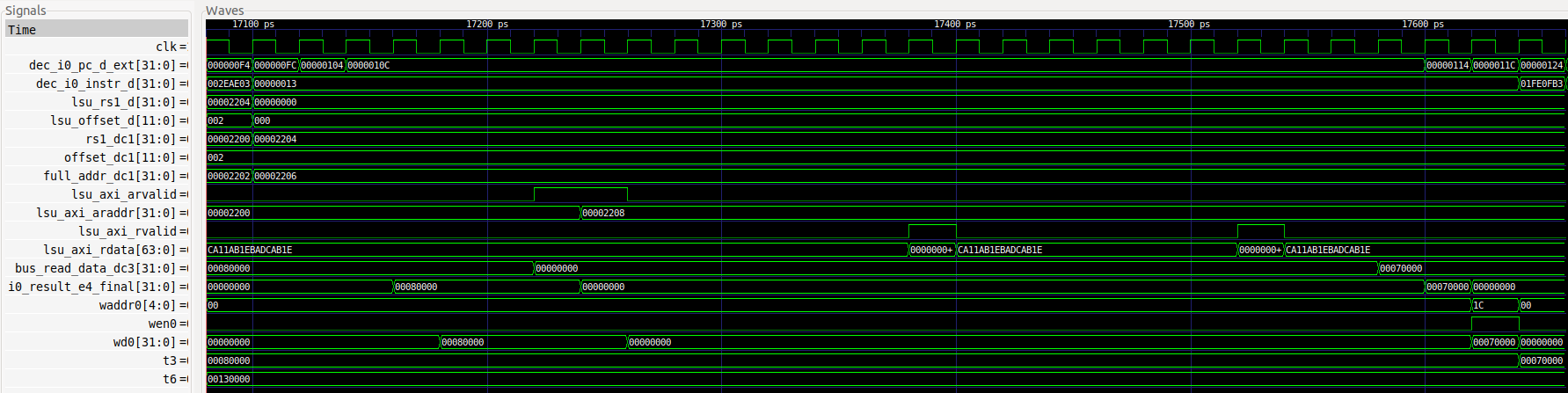
**TAREFA:** Analisar as escritas não alinhadas na DCCM, bem como escritas de sub-palavras: store byte (sb) ou *store half-word* (sh).

Solução não fornecida.

**TAREFA:** Replique a simulação da Figura 17 no seu computador. Use o ficheiro *test\_Blocking.tcl* (fornecido em *[RVfpgaPath]/RVfpga/Labs/Lab13/LW\_Instruction\_ExtMemory*). Aumente o *zoom* ( ) várias vezes e passe para 16940ps.

Solução fornecida no documento principal do Lab 13.

**TAREFA:** Modifique o programa da Figura 15 para analisar um acesso de leitura não alinhado que precisa enviar dois endereços para a memória externa por meio do barramento AXI.

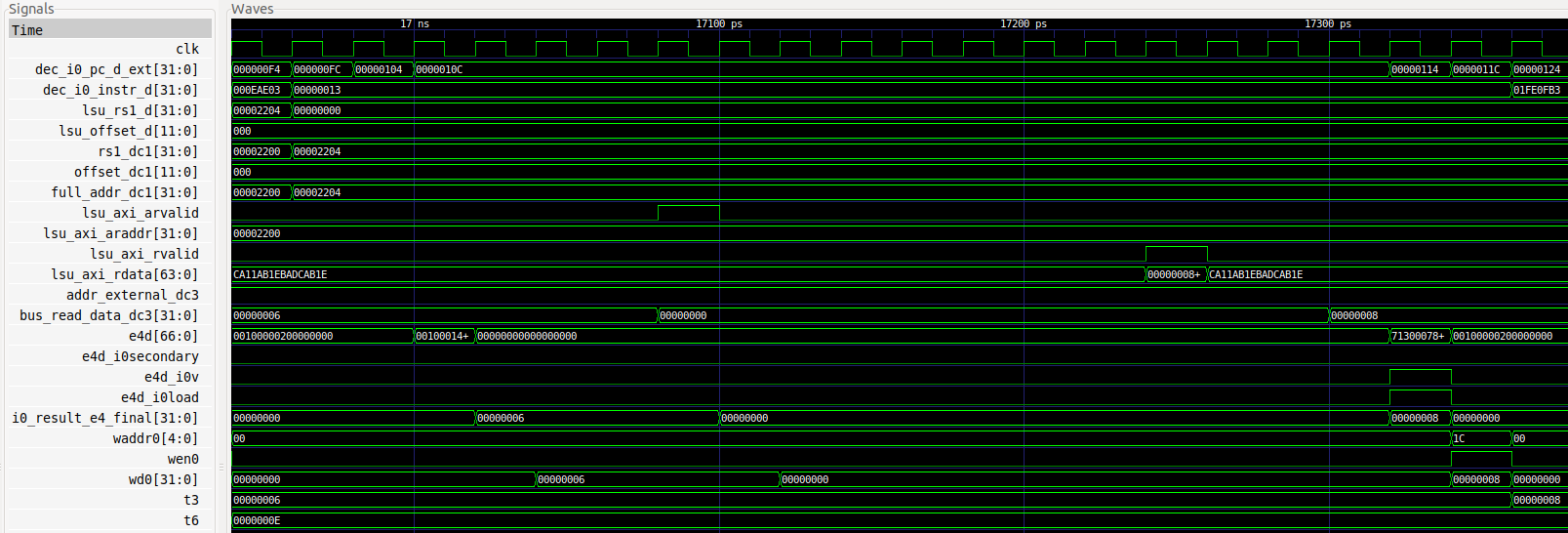


**TAREFA**: Adicione à simulação os sinais que controlam os multiplexers (nos andares DC3 e Commit na Figura 16) que selecionam os dados fornecidos pela memória externa DDR. pode encontrar esses multiplexers nas seguintes linhas do código Verilog:

- Multiplexador 2:1: Linha 264 do módulo **lsu\_lsc\_ctl**.

- Multiplexador 3:1: Linha 2277 do módulo **dec\_decode\_ctl**.

Um ficheiro *.tcl* que pode usar é fornecido em: *[RVfpgaPath]/RVfpga/Labs/Lab13/LW\_Instruction\_ExtMemory/test\_Blocking\_Extended.tcl*



**TAREFA:** Também pode ser interessante analisar a implementação do barramento AXI para aceder ao controlador DRAM, para o qual pode inspecionar o módulo **lsu\_bus\_intf**.

Solução não fornecida.

**TAREFA:** Replique a simulação da Figura 18 no seu computador. Use o ficheiro *scriptStoreBuffer.tcl* (fornecido em *[RVfpgaPath]/RVfpga/Labs/Lab13/SW\_Instruction\_DCCM*). Aumente o *zoom* ( ) várias vezes e passe para 17900ps.

Solução fornecida no documento principal do Lab 13.

**TAREFA:** Modifique o programa da Figura 11 para ter duas escritas pendentes e realizar uma análise semelhante à da Figura 18.

Solução não fornecida.