

## 1. EXERCÍCIOS

### Exercício 1

As soluções completas para este exercício são fornecidas na pasta:

*[RVfpgaPath]/RVfpga/Labs/RVfpgaLabsSolutions/Programs\_Solutions/Lab18/Minu*

### Exercício 2

As soluções completas para este exercício são fornecidas na pasta:

*[RVfpgaPath]/RVfpga/Labs/RVfpgaLabsSolutions/Programs\_Solutions/Lab18/MinMaxMinuMaxu*

### Exercícios 3 e 4

As soluções completas para estes dois exercícios são fornecidas na pasta:

*[RVfpgaPath]/RVfpga/Labs/RVfpgaLabsSolutions/Programs\_Solutions/Lab18/FloatingPoint*

### Exercícios 5 e 6

Soluções não fornecidas.

### Exercício 7

As soluções completas para este exercício são fornecidas na pasta:

*[RVfpgaPath]/RVfpga/Labs/RVfpgaLabsSolutions/Programs\_Solutions/Lab18/NewHwCounter*

a. **Alterações no código Verilog:**

i. Ficheiro **swerv\_types.sv**:

```
29  typedef enum logic [3:0] {
30      NULL      = 4'b0000,
31      MUL       = 4'b0001,
32      LOAD      = 4'b0010,
33      STORE     = 4'b0011,
34      ALU       = 4'b0100,
35      CSRREAD   = 4'b0101,
36      CSRWRITE  = 4'b0110,
37      CSRRW     = 4'b0111,
38      EBREAK    = 4'b1000,
39      ECALL     = 4'b1001,
40      FENCE     = 4'b1010,
41      FENCEI    = 4'b1011,
42      MRET      = 4'b1100,
43      CONDBR    = 4'b1101,
44      JAL       = 4'b1110,
45      // New HW-Counter
46      IMM       = 4'b1111
47  } inst_t;
48
```

ii. Ficheiro **dec\_tlu.ctl.sv**:

```

1926 `define MHPME_BR_ERROR          6'd41
1927 `define MHPME_IBUS_TRANS         6'd42 // OOP
1928 `define MHPME_DBUS_TRANS         6'd43 // OOP
1929 `define MHPME_DBUS_MA_TRANS      6'd44 // OOP
1930 `define MHPME_IBUS_ERROR         6'd45 // OOP
1931 `define MHPME_DBUS_ERROR         6'd46 // OOP
1932 `define MHPME_IBUS_STALL         6'd47 // OOP
1933 `define MHPME_DBUS_STALL         6'd48 // OOP
1934 `define MHPME_INT_DISABLED       6'd49 // OOP
1935 `define MHPME_INT_STALLED        6'd50 // OOP
1936 // New HW-Counter
1937 `define MHPME_INST_IMM           6'd51 // OOP

```

```

1963 // Generate the muxed incs for all counters based on event type
1964 for (genvar i=0 ; i < 4; i++) begin
1965     assign mhpme_inc_e4[i][1:0] = {2{mhpme}} &
1966     (
1967         ({2{(mhpme_vec[i][5:0] == `MHPME_CLK_ACTIVE)}} & 2'b01) |
1968         ({2{(mhpme_vec[i][5:0] == `MHPME_ICACHE_HIT)}} & {1'b0, ifu_pmu_ic_hit}) |
1969         ({2{(mhpme_vec[i][5:0] == `MHPME_ICACHE_MISS)}} & {1'b0, ifu_pmu_ic_miss}) |
1970         ({2{(mhpme_vec[i][5:0] == `MHPME_INST_COMMIT)}} & {tlu_i0_commit_cmt, tlu_i0_commit_cmt & ~illegal_e4}) |
1971         ({2{(mhpme_vec[i][5:0] == `MHPME_INST_COMMIT_16B)}} & {tlu_i0_commit_cmt & ~exu_pmu_i0_pc4, tlu_i0_commit_cmt & ~exu_pmu_i0_pc4 & ~il
1972         ({2{(mhpme_vec[i][5:0] == `MHPME_INST_COMMIT_32B)}} & {tlu_i0_commit_cmt & exu_pmu_i0_pc4, tlu_i0_commit_cmt & exu_pmu_i0_pc4 & ~il
1973         ({2{(mhpme_vec[i][5:0] == `MHPME_INST_ALIGNED)}} & ifu_pmu_instr_aligned[1:0]) |
1974         ({2{(mhpme_vec[i][5:0] == `MHPME_INST_DECODED)}} & dec_pmu_instr_decoded[1:0]) |
1975         ({2{(mhpme_vec[i][5:0] == `MHPME_ALGMR_STALL)}} & {1'b0,ifu_pmu_align_stall}) |
1976         ({2{(mhpme_vec[i][5:0] == `MHPME_DECODE_STALL)}} & {1'b0,dec_pmu_decode_stall}) |
1977         ({2{(mhpme_vec[i][5:0] == `MHPME_INST_MUL)}} & {(pmu_i0_itype_qual[3:0] == MUL), (pmu_i0_itype_qual[3:0] == MUL)}) |
1978         ({2{(mhpme_vec[i][5:0] == `MHPME_INST_DIV)}} & {1'b0,dec_tlu_packet_e4.pmu_divide & tlu_i0_commit_cmt}) |
1979         ({2{(mhpme_vec[i][5:0] == `MHPME_INST_LOAD)}} & {(pmu_i0_itype_qual[3:0] == LOAD), (pmu_i0_itype_qual[3:0] == LOAD)}) |
1980         ({2{(mhpme_vec[i][5:0] == `MHPME_INST_STORE)}} & {(pmu_i0_itype_qual[3:0] == STORE), (pmu_i0_itype_qual[3:0] == STORE)}) |
1981         ({2{(mhpme_vec[i][5:0] == `MHPME_INST_MALOAD)}} & {(pmu_i0_itype_qual[3:0] == LOAD), (pmu_i0_itype_qual[3:0] == LOAD)} &
1982             {2{dec_tlu_packet_e4.pmu_lsu_misaligned}}) |
1983         ({2{(mhpme_vec[i][5:0] == `MHPME_INST_MASTORE)}} & {(pmu_i0_itype_qual[3:0] == STORE), (pmu_i0_itype_qual[3:0] == STORE)} &
1984             {2{dec_tlu_packet_e4.pmu_lsu_misaligned}}) |
1985         ({2{(mhpme_vec[i][5:0] == `MHPME_INST_ALU)}} & {(pmu_i0_itype_qual[3:0] == ALU), (pmu_i0_itype_qual[3:0] == ALU)}) |
1986         // New HW-Counter
1987         ({2{(mhpme_vec[i][5:0] == `MHPME_INST_IMM)}} & {(pmu_i0_itype_qual[3:0] == IMM), (pmu_i0_itype_qual[3:0] == IMM)}) |
1988         ({2{(mhpme_vec[i][5:0] == `MHPME_INST_SRRREAD)}} & {1'b0, (pmu_i0_itype_qual[3:0] == SRRREAD)}) |

```

```

2111 // -----
2112 // MHPME3(RW)
2113 // [5:0] : Hardware Performance Monitor Event 3
2114 `define MHPME3 12'h323
2115
2116 // we only have 50 events, HPME* are WARL so saturate at 50
2117 logic [5:0] event_saturate_wb;
2118 // assign event_saturate_wb[5:0] = ((dec_csr_wdata_wb[5:0] > 6'd50) | (|dec_csr_wdata_wb[31:6])) ? 6'd50 : dec_csr_wdata_wb[5:0];
2119 // New HW-Counter
2120 assign event_saturate_wb[5:0] = ((dec_csr_wdata_wb[5:0] > 6'd51) | (|dec_csr_wdata_wb[31:6])) ? 6'd51 : dec_csr_wdata_wb[5:0];
2121
2122 assign wr_mhpme3_wb = dec_csr_wen_wb_mod & (dec_csr_wraddr_wb[11:0] == `MHPME3);
2123 rvdffs #(6) mhpme3_ff (.*, .clk(active_clk), .en(wr_mhpme3_wb), .din(event_saturate_wb[5:0]), .dout(mhpme3[5:0]));
2124 // -----

```

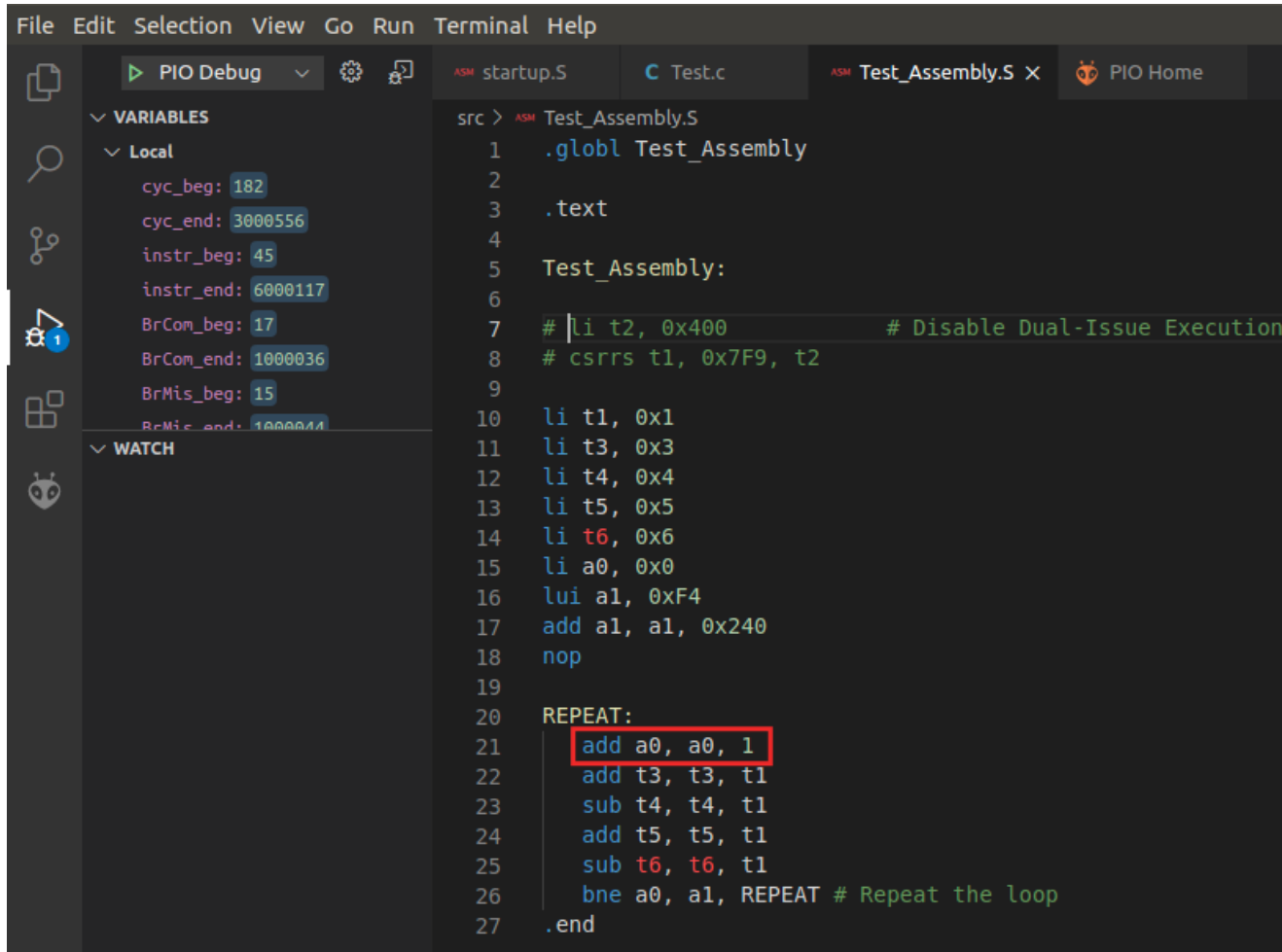
iii. Ficheiro **dec\_decode\_ctl.sv**:

```

952 // the classes must be mutually exclusive with one another
953
954 always_comb begin
955     i0_itype = NULL;
956     i1_itype = NULL;
957
958     if (i0_legal_decode_d) begin
959         if (i0_dp.mul)          i0_itype = MUL;
960         if (i0_dp.load)        i0_itype = LOAD;
961         if (i0_dp.store)       i0_itype = STORE;
962         if (i0_dp.pm_alu)       i0_itype = ALU;
963         // New HW-Counter
964         if (i0_dp.imm12 & i0_dp.pm_alu) i0_itype = IMM;
965         if (csr_read & ~csr_write) i0_itype = CSRREAD;
966         if (~csr_read & csr_write) i0_itype = CSRWRITE;
967         if (csr_read & csr_write) i0_itype = CSRRW;
968         if (i0_dp.ebreak)        i0_itype = EBREAK;
969         if (i0_dp.ecall)         i0_itype = ECALL;
970         if (i0_dp.fence)         i0_itype = FENCE;
971         if (i0_dp.fence_i)       i0_itype = FENCEI;
972         if (i0_dp.mret)          i0_itype = MRET;
973         if (i0_dp.condb_r)       i0_itype = CONDBR;
974         if (i0_dp.jal)           i0_itype = JAL;
975     end
976
977     if (dec_i1_decode_d) begin
978         if (i1_dp.mul)          i1_itype = MUL;
979         if (i1_dp.load)        i1_itype = LOAD;
980         if (i1_dp.store)       i1_itype = STORE;
981         if (i1_dp.pm_alu)       i1_itype = ALU;
982         // New HW-Counter
983         if (i1_dp.imm12 & i1_dp.pm_alu) i1_itype = IMM;
984         if (i1_dp.condb_r)       i1_itype = CONDBR;
985         if (i1_dp.jal)           i1_itype = JAL;
986     end
987

```

## b. Simulação Verilator:



The screenshot shows the PIO Debug IDE interface. On the left, there is a sidebar with icons for file explorer, search, and a list of variables and watch points. The main area displays assembly code for 'Test\_Assembly.S'.

**VARIABLES:**

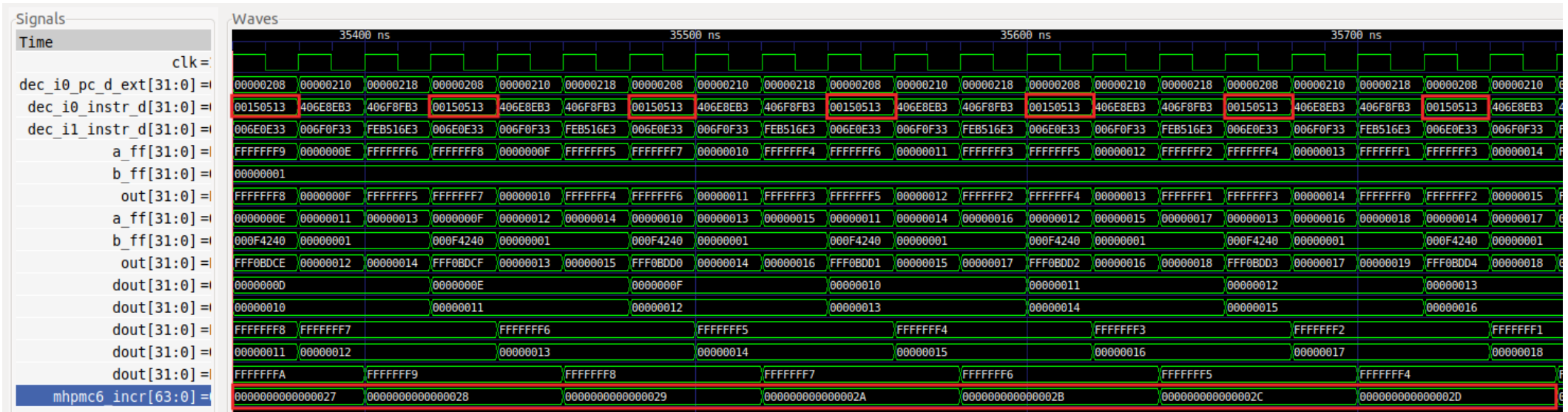
- Local
  - cyc\_beg: 182
  - cyc\_end: 3000556
  - instr\_beg: 45
  - instr\_end: 6000117
  - BrCom\_beg: 17
  - BrCom\_end: 1000036
  - BrMis\_beg: 15
  - BrMis\_end: 1000044

**WATCH:**

The main editor shows the following assembly code:

```
src > ASM Test_Assembly.S
1  .globl Test_Assembly
2
3  .text
4
5  Test_Assembly:
6
7  # li t2, 0x400          # Disable Dual-Issue Execution
8  # csrfs t1, 0x7F9, t2
9
10 li t1, 0x1
11 li t3, 0x3
12 li t4, 0x4
13 li t5, 0x5
14 li t6, 0x6
15 li a0, 0x0
16 lui a1, 0xF4
17 add a1, a1, 0x240
18 nop
19
20 REPEAT:
21   add a0, a0, 1
22   add t3, t3, t1
23   sub t4, t4, t1
24   add t5, t5, t1
25   sub t6, t6, t1
26   bne a0, a1, REPEAT # Repeat the loop
27 .end
```

```
src > C Test.c > ...
19  /* Initialize UART */
20  uartInit();
21
22  pspEnableAllPerformanceMonitor(1);
23
24  pspPerformanceCounterSet(D_PSP_COUNTER0, E_CYCLES_CLOCKS_ACTIVE);
25  pspPerformanceCounterSet(D_PSP_COUNTER1, E_INSTR_COMMITTED_ALL);
26  pspPerformanceCounterSet(D_PSP_COUNTER2, E_BRANCHES_COMMITTED);
27  pspPerformanceCounterSet(D_PSP_COUNTER3, 51);
28
29  cyc_beg = pspPerformanceCounterGet(D_PSP_COUNTER0);
30  instr_beg = pspPerformanceCounterGet(D_PSP_COUNTER1);
31  BrCom_beg = pspPerformanceCounterGet(D_PSP_COUNTER2);
32  BrMis_beg = pspPerformanceCounterGet(D_PSP_COUNTER3);
33
34  Test_Assembly();
35
36  cyc_end = pspPerformanceCounterGet(D_PSP_COUNTER0);
37  instr_end = pspPerformanceCounterGet(D_PSP_COUNTER1);
38  BrCom_end = pspPerformanceCounterGet(D_PSP_COUNTER2);
39  BrMis_end = pspPerformanceCounterGet(D_PSP_COUNTER3);
40
41  printfNexys("Cycles = %d", cyc_end-cyc_beg);
42  printfNexys("Instructions = %d", instr_end-instr_beg);
43  printfNexys("BrCom = %d", BrCom_end-BrCom_beg);
44  printfNexys("12-bit Immediate Instructions = %d", BrMis_end-BrMis_beg);
45
46  while(1);
47 }
```



c. Execução na placa:

```
platformio.ini  ASM startup.S  C Test.c  PIO Home  ASM Test_Assembly.S x
src > ASM Test_Assembly.S
3  .text
4
5  Test_Assembly:
6
7  # li t2, 0x400          # Disable Dual-Issue Execution
8  # csrrs t1, 0x7F9, t2
9
10 li t1, 0x1
11 li t3, 0x3
12 li t4, 0x4
13 li t5, 0x5
14 li t6, 0x6
15 li a0, 0x0
16 li a1, 1000
17 nop
18
19 REPEAT:
20 add a0, a0, 1
21 add t3, t3, 2
22 sub t4, t4, t1
23 add t5, t5, t1
24 sub t6, t6, t1
25 bne a0, a1, REPEAT # Repeat the loop
26 .end

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

> Executing task: platformio device monitor <

--- Available filters and text transformations: colorize, debug, default, direct,
--- More details at https://bit.ly/pio-monitor-filters
--- Miniterm on /dev/ttyUSB1 115200,8,N,1 ---
--- Quit: Ctrl+C | Menu: Ctrl+T | Help: Ctrl+T followed by Ctrl+H ---
Cycles = 3413
Instructions = 6071
BrCom = 1019
12-bit Immediate Instructions = 2029
```