Instruções de uso – MIPS Single Cycle

O projeto entregue agora é um MIPS Single Cycle com as instruções do grupo A e B implementadas. Sendo que elas englobam:

Grupo A:

- As instruções de que interagem com a memória, seja para escrever ou ler:
  - Carrega palavra (*load word: lw*);
  - Armazena palavra (store word: sw).
- As instruções que realizam alguma operação lógica ou algébrica na ULA:
  - Soma (add);
  - Subtração (sub);
  - E lógico (AND);
  - OU lógico (OR);
  - Comparação menor que (set if less than: slt).
- As instruções de desvio:
  - Desvio se igual (branch equal: beq);
  - Salto incondicional (jump: j).

Grupo B:

- A instrução de carga:
  - Carrega imediato para 16 bits MSB (load upper immediate: lui).
- As instruções que realizam alguma operação lógica ou algébrica na ULA:
  - Soma com imediato (*addi*);
  - E lógico com imediato (*ANDI*);
  - OU lógico com imediato (*ORI*);
  - Comparação menor que imediato (*set if less than: slti*)
- As instruções de desvio:
  - Desvio se não igual (*branch not equal: bne*);
  - Salto e conecta (*jump and link: jal*);
  - Salto por registrador (*jump register: jr*).

Para se testar todas essas funções, nos foi disponibilizado uma sequência de instruções para se colocar na ROM:

```
    -- Valores iniciais no banco de registradores:
    -- $t0   (#8) := 0x00
    -- $t1   (#9) := 0x0A
    -- $t2  (#10) := 0x0B
    -- $t3  (#11) := 0x0C
    -- $t4  (#12) := 0x0D
    -- $t5  (#13) := 0x16

 0    :   AC090008; --sw $t1 8($zero)      (m(8) := 0x0000000A)
 1    :   8C080008; --lw $t0 8($zero)      ($t0 := 0x0000000A)
 2    :   012A4022; --sub $t0 $t1 $t2      ($t0 := 0xFFFFFFFF)
 3    :   012A4024; --and $t0 $t1 $t2      ($t0 := 0x0000000A)
```

```
    4    :    012A4025; --or $t0 $t1 $t2       ($t0 := 0x0000000B)
    5    :    3C08FFFF; --lui $t0 0xFFFF       ($t0 := 0xFFFF0000)
    6    :    2128000A; --addi $t0 $t1 0x000A  ($t0 := 0x00000014)
    7    :    31080013; --andi $t0 $t0 0x0013  ($t0 := 0x00000010)
    8    :    35880007; --ori $t0 $t4 0x0007   ($t0 := 0x0000000F)
    9    :    2928FFFF; --slti $t0 $t1 0xFFFF  ($t0 := 0x00000000)
    10   :    010A4020; --add $t0 $t0 $t2      ($t0 := 0x0000000B)
                           --segunda execução ($t0 := 0x00000016)
    11   :    150DFFFE; --bne $t0 $t5 0xFFFE   (pc := #10)
                           --segunda execução (pc := #12)
    12   :    012A402A; --slt $t0 $t1 $t2      ($t0 := 0x00000001)
    13   :    010A4020; --add $t0 $t0 $t2      ($t0 := 0x0000000C)
                           --segunda execução ($t0 := 0x00000017)
    14   :    110BFFFE; --beq $t0 $t3 0xFFFE   (pc := #13)
                           --segunda execução (pc := #15)
    15   :    0C00001F; --jal 0x00001F         (pc := #31)
    17   :    08000000; --j 0x000000           (pc := #0)
    31   :    03E00008; --jr $ra               (pc := #17)
END;
```

Ao ligar a placa já com o código todo compilado, os displays de sete segmentos servirão como forma de checar a saída. Por padrão eles irão mostrar o PC (Program Counter) da ROM. Com o programa rodando, basta segurar o botão KEY(1) para mudar o que os displays mostram para o valor da saída da ULA. Por último, para avançar o clock, a instrução que esta sendo executada, basta apertar o botão KEY(1).