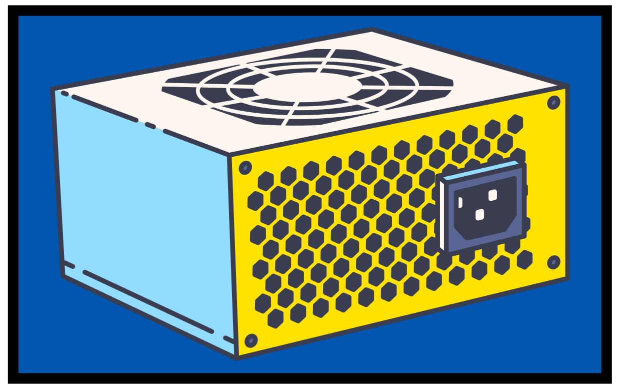
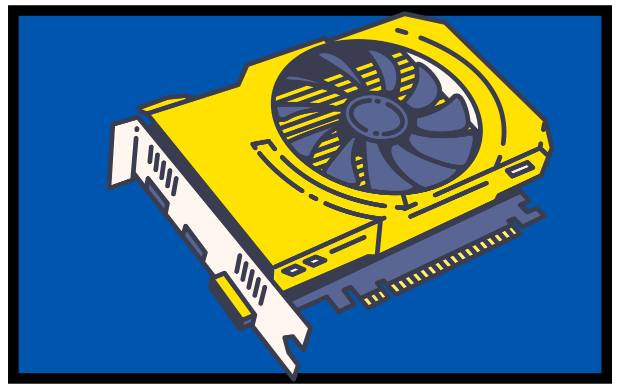
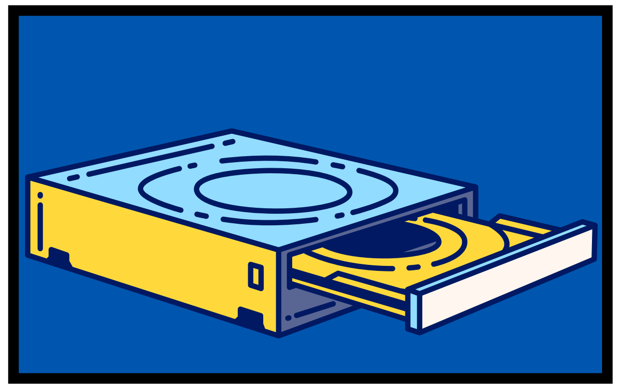
PROCESSADOR ARQUITETURA



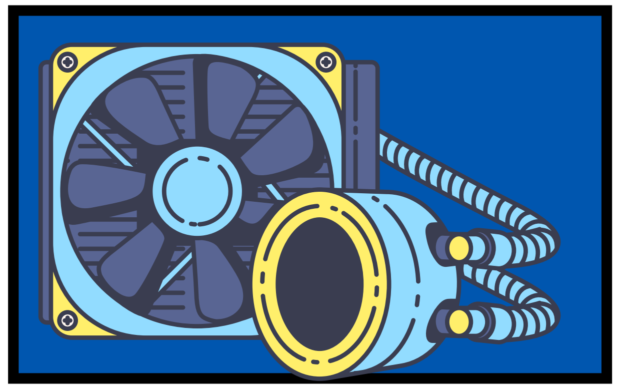
MARIA EDUARDA PEDROSO



CAIO VINICIUS



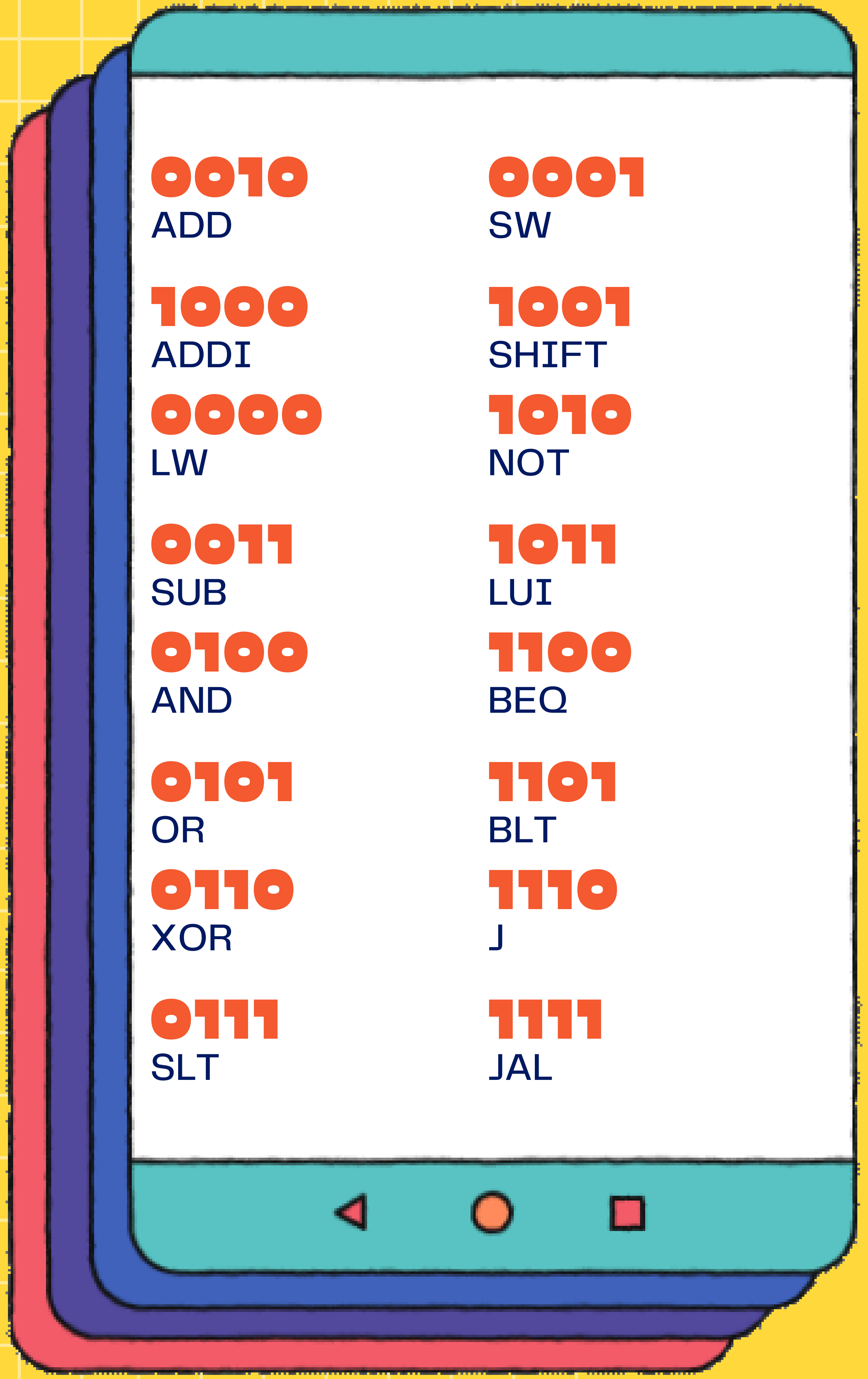
GUILHERME KATO



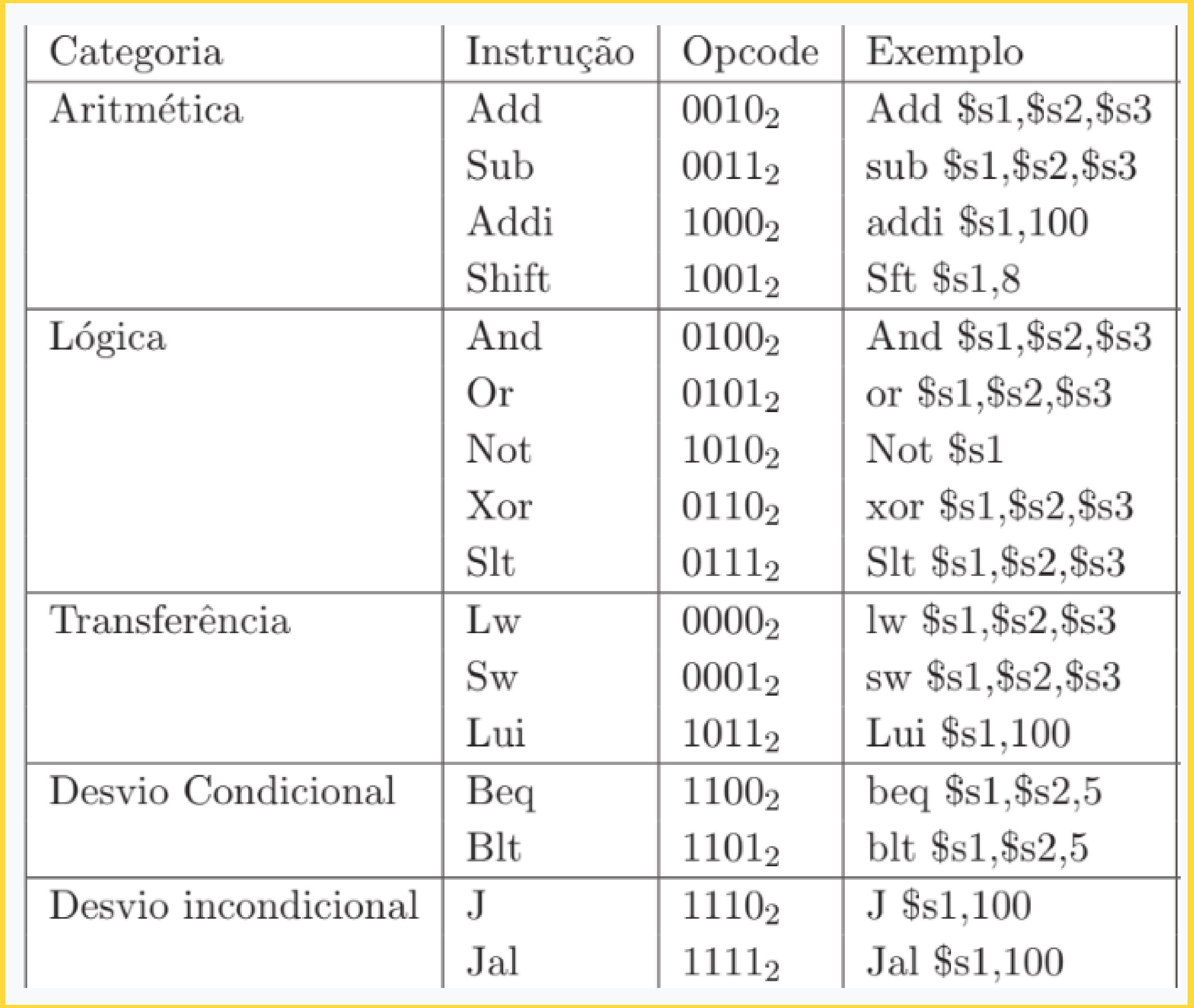
RODRIGO OLIVEIRA

PROCESSADOR

upcodes

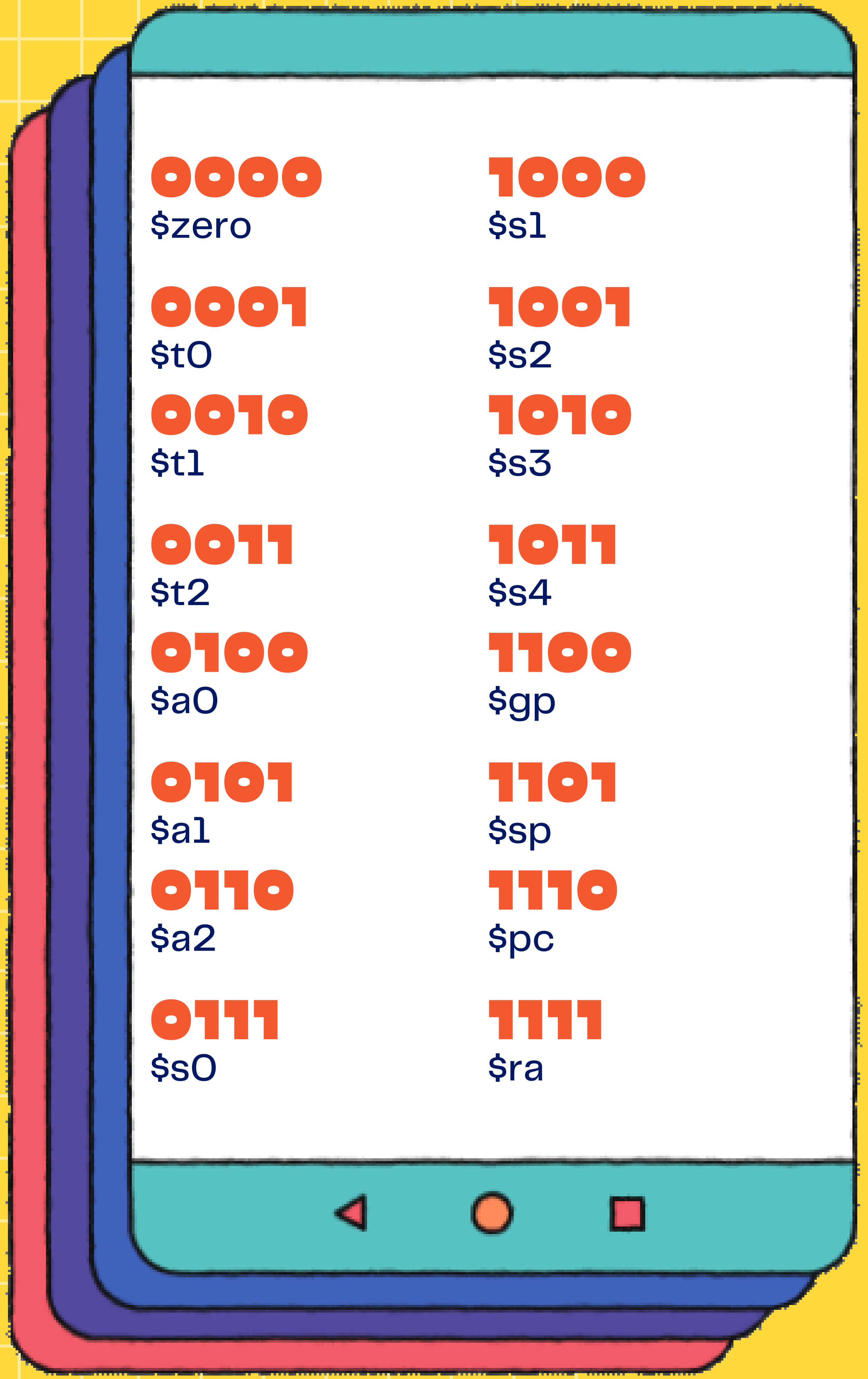


0010 ADD	0001 SW
1000 ADDI	1001 SHIFT
0000 LW	1010 NOT
0011 SUB	1011 LUI
0100 AND	1100 BEQ
0101 OR	1101 BLT
0110 XOR	1110 J
0111 SLT	1111 JAL

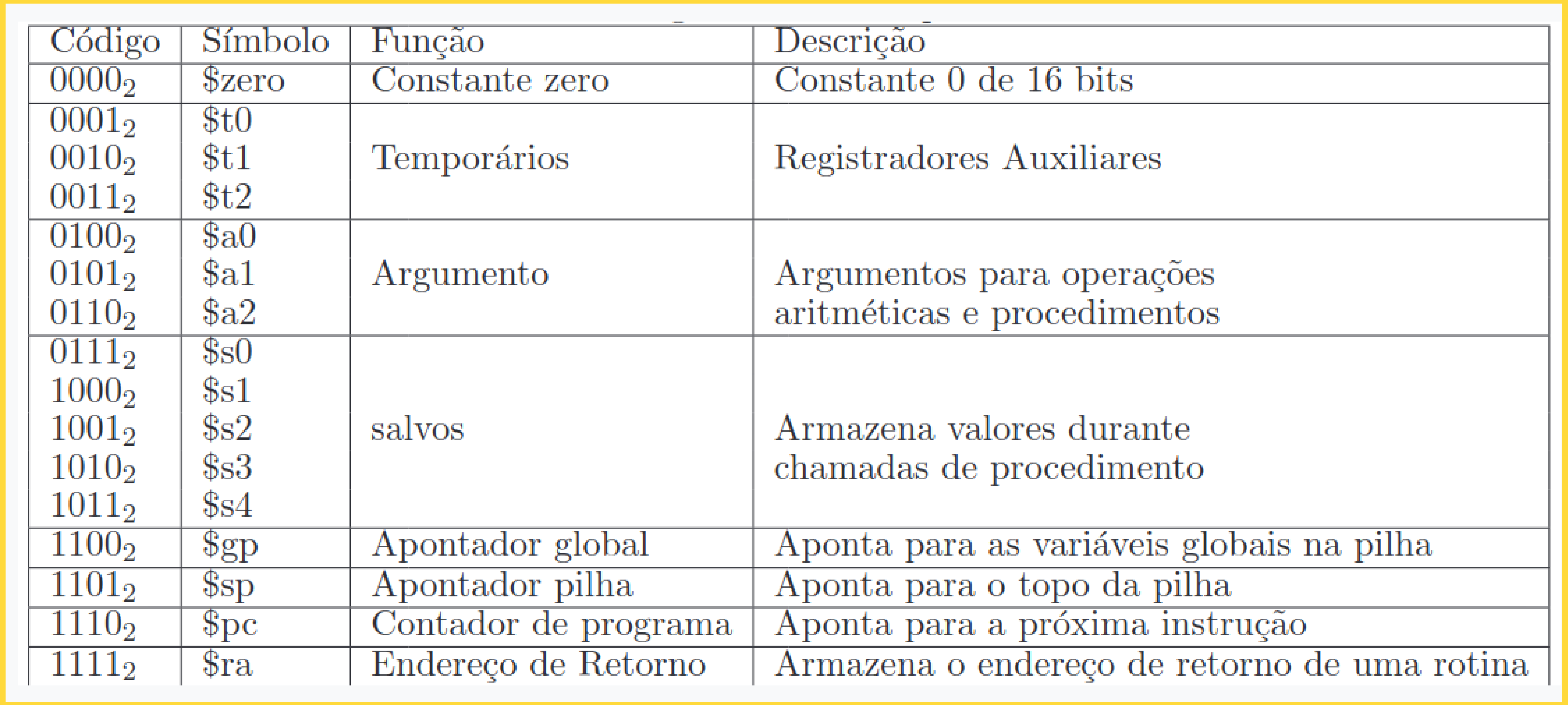


Categoria	Instrução	Opcode	Exemplo
Aritmética	Add	0010_2	Add \$s1,\$s2,\$s3
	Sub	0011_2	sub \$s1,\$s2,\$s3
	Addi	1000_2	addi \$s1,100
	Shift	1001_2	Sft \$s1,8
Lógica	And	0100_2	And \$s1,\$s2,\$s3
	Or	0101_2	or \$s1,\$s2,\$s3
	Not	1010_2	Not \$s1
	Xor	0110_2	xor \$s1,\$s2,\$s3
	Slt	0111_2	Slt \$s1,\$s2,\$s3
Transferência	Lw	0000_2	lw \$s1,\$s2,\$s3
	Sw	0001_2	sw \$s1,\$s2,\$s3
	Lui	1011_2	Lui \$s1,100
Desvio Condicional	Beq	1100_2	beq \$s1,\$s2,5
	Blt	1101_2	blt \$s1,\$s2,5
Desvio incondicional	J	1110_2	J \$s1,100
	Jal	1111_2	Jal \$s1,100

register

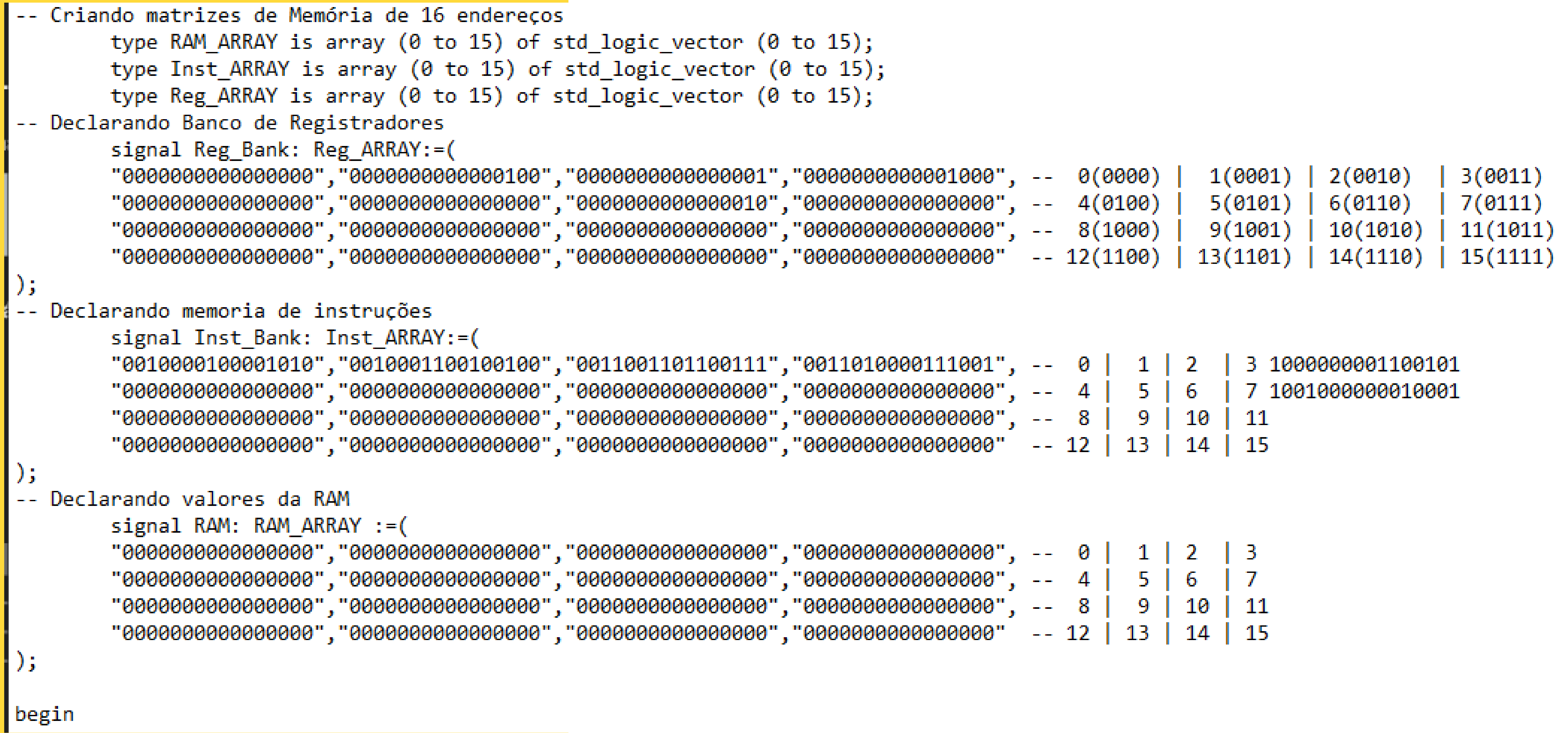


0000	1000
$zero	$s1
0001	1001
$t0	$s2
0010	1010
$t1	$s3
0011	1011
$t2	$s4
0100	1100
$a0	$gp
0101	1101
$a1	$sp
0110	1110
$a2	$pc
0111	1111
$s0	$ra



Código	Símbolo	Função	Descrição
0000_2	$zero	Constante zero	Constante 0 de 16 bits
0001_2	$t0	Temporários	Registradores Auxiliares
0010_2	$t1		
0011_2	$t2		
0100_2	$a0	Argumento	Argumentos para operações aritméticas e procedimentos
0101_2	$a1		
0110_2	$a2		
0111_2	$s0	salvos	Armazena valores durante chamadas de procedimento
1000_2	$s1		
1001_2	$s2		
1010_2	$s3		
1011_2	$s4		
1100_2	$gp	Apontador global	Aponta para as variáveis globais na pilha
1101_2	$sp	Apontador pilha	Aponta para o topo da pilha
1110_2	$pc	Contador de programa	Aponta para a próxima instrução
1111_2	$ra	Endereço de Retorno	Armazena o endereço de retorno de uma rotina

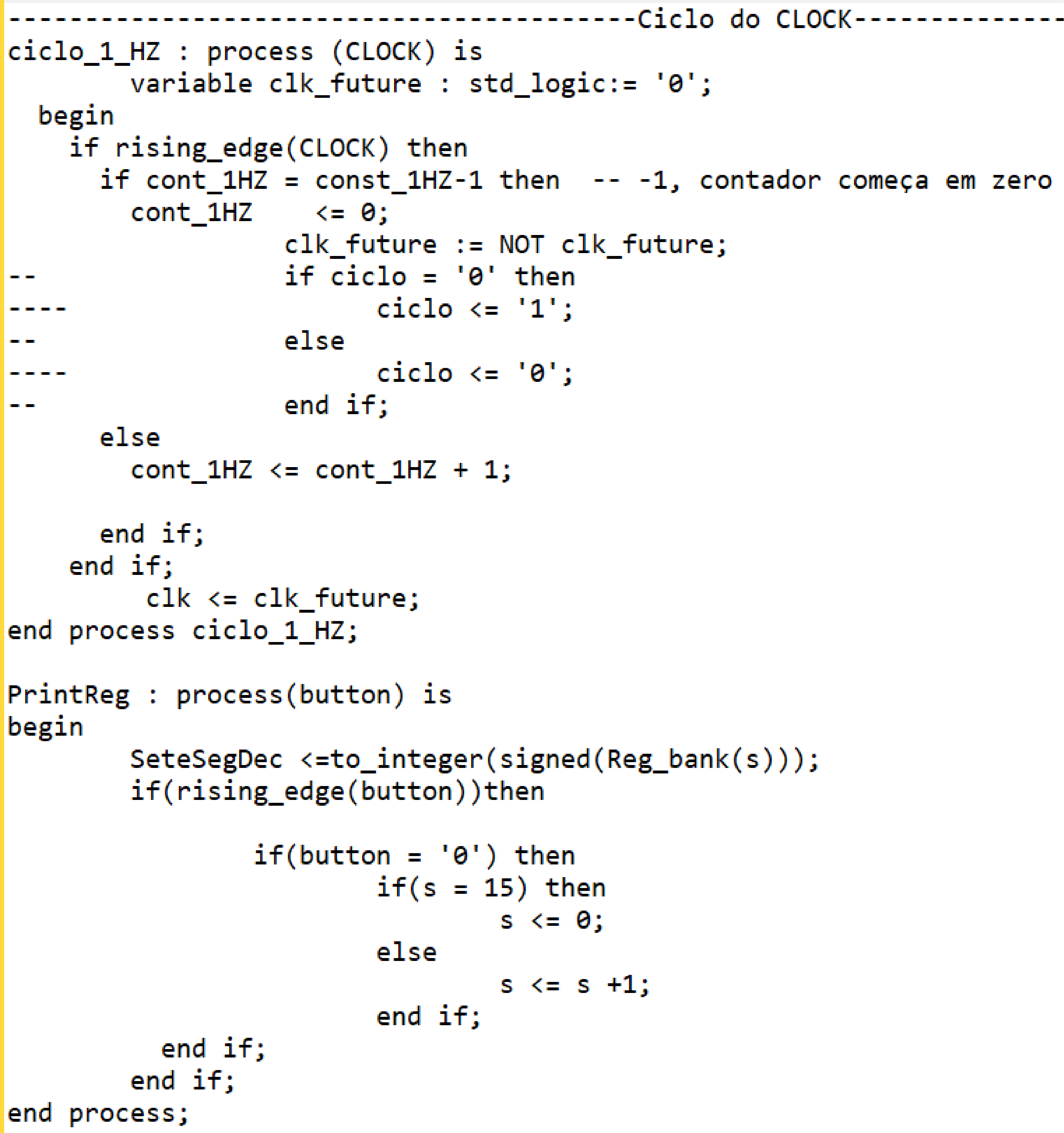
Registrar



```
-- Criando matrizes de Memória de 16 endereços
        type RAM_ARRAY is array (0 to 15) of std_logic_vector (0 to 15);
        type Inst_ARRAY is array (0 to 15) of std_logic_vector (0 to 15);
        type Reg_ARRAY is array (0 to 15) of std_logic_vector (0 to 15);
-- Declarando Banco de Registradores
        signal Reg_Bank: Reg_ARRAY:=(
        "0000000000000000","0000000000000100","0000000000000001","0000000000001000", --  0(0000) |  1(0001) | 2(0010)  | 3(0011)
        "0000000000000000","0000000000000000","0000000000000010","0000000000000000", --  4(0100) |  5(0101) | 6(0110)  | 7(0111)
        "0000000000000000","0000000000000000","0000000000000000","0000000000000000", --  8(1000) |  9(1001) | 10(1010) | 11(1011)
        "0000000000000000","0000000000000000","0000000000000000","0000000000000000"  -- 12(1100) | 13(1101) | 14(1110) | 15(1111)
);
-- Declarando memoria de instruções
        signal Inst_Bank: Inst_ARRAY:=(
        "0010000100001010","0010001100100100","0011001101100111","0011010000111001", --  0 |  1 | 2  | 3 1000000001100101
        "0000000000000000","0000000000000000","0000000000000000","0000000000000000", --  4 |  5 | 6  | 7 1001000000010001
        "0000000000000000","0000000000000000","0000000000000000","0000000000000000", --  8 |  9 | 10 | 11
        "0000000000000000","0000000000000000","0000000000000000","0000000000000000"  -- 12 | 13 | 14 | 15
);
-- Declarando valores da RAM
        signal RAM: RAM_ARRAY :=(
        "0000000000000000","0000000000000000","0000000000000000","0000000000000000", --  0 |  1 | 2  | 3
        "0000000000000000","0000000000000000","0000000000000000","0000000000000000", --  4 |  5 | 6  | 7
        "0000000000000000","0000000000000000","0000000000000000","0000000000000000", --  8 |  9 | 10 | 11
        "0000000000000000","0000000000000000","0000000000000000","0000000000000000"  -- 12 | 13 | 14 | 15
);

begin
```

Ciclo clock



```
--------------------Ciclo do CLOCK-------
ciclo_1_HZ : process (CLOCK) is
        variable clk_future : std_logic:= '0';
  begin
    if rising_edge(CLOCK) then
      if cont_1HZ = const_1HZ-1 then  -- -1, contador começa em zero
        cont_1HZ    <= 0;
                  clk_future := NOT clk_future;
--                if ciclo = '0' then
----                    ciclo <= '1';
--                else
----                    ciclo <= '0';
--                end if;
      else
        cont_1HZ <= cont_1HZ + 1;

      end if;
    end if;
         clk <= clk_future;
end process ciclo_1_HZ;

PrintReg : process(button) is
begin
        SeteSegDec <=to_integer(signed(Reg_bank(s)));
        if(rising_edge(button))then

                if(button = '0') then
                        if(s = 15) then
                                s <= 0;
                        else
                                s <= s +1;
                        end if;
          end if;
        end if;
end process;
```

ADD

```
when "0010" =>
Reg_bank(to_integer(unsigned(instruct(12 to 15)))) <=
 Reg_bank(to_integer(unsigned(instruct(4 to 7)))) +
Reg_bank(to_integer(unsigned(instruct(8 to 11))));
 destino <= to_integer(unsigned(instruct(12 to 15)));
DECOP2 <= "0001000"; -- 'A' de add;
```

SUB

```
when "0011" =>
Reg_bank(to_integer(unsigned(instruct(12 to 15)))) <=
Reg_bank(to_integer(unsigned(instruct(4 to 7)))) -
Reg_bank(to_integer(unsigned(instruct(8 to 11))));
destino <= to_integer(unsigned(instruct(12 to 15)));
DECOP2 <= "0100100"; -- 'S' de sub;
```

ADDI

```
when "1000" =>
 Reg_bank(to_integer(unsigned(instruct(4 to 7)))) <=

std_logic_vector(to_unsigned(to_integer(unsigned(in
struct(8 to 15))), 16));
DECOP2 <= "1001001"; -- 'A' de addi;
 Reg_bank(14) <= Reg_bank(14) +
std_logic_vector(to_unsigned(1,16));
```

SHIFT

```
when "1001" =>
Reg_bank(to_integer(unsigned(instruct(8 to 11)))) <=

std_logic_vector(shift_left(unsigned(Reg_bank(to_inte
ger(unsigned(instruct(8 to
11))))),to_integer(unsigned(instruct(12 to 15)))));
destino <= to_integer(unsigned(instruct(8 to 11)));
DECOP2 <= "011100"; -- 'S' de shift;
```

PROCESSADOR LÓGICA

AND

```
when "0100" =>
Reg_bank(to_integer(unsigned(instruct(12 to
15)))) <=
Reg_bank(to_integer(unsigned(instruct(4 to
7)))) AND
Reg_bank(to_integer(unsigned(instruct(8 to 11))));
DECOP2 <= "0110000"; -- 'E' de and E
Reg_bank(14) <= Reg_bank(14) +
std_logic_vector(to_unsigned(1,16));
```

OR

```
when "0101" =>
Reg_bank(to_integer(unsigned(instruct(12
to 15)))) <=
Reg_bank(to_integer(unsigned(instruct(4
to 7)))) OR
Reg_bank(to_integer(unsigned(instruct(8 to
11))));
DECOP2 <= "0100100"; -- '5' de OR
Reg_bank(14) <= Reg_bank(14) +
std_logic_vector(to_unsigned(1,16));
```

XOR

```
when "0110" =>
Reg_bank(to_integer(unsigned(instruct(12 to 15)))) <=

Reg_bank(to_integer(unsigned(instruct(4 to 7)))) XOR
Reg_bank(to_integer(unsigned(instruct(8 to 11))));
DECOP2 <= "0100000"; -- '6' de XOR
Reg_bank(14) <= Reg_bank(14) +
std_logic_vector(to_unsigned(1,16));
```

SLT

```
when "0111" =>
if((Reg_bank(to_integer(unsigned(instruct(4 to 7))))) <
(Reg_bank(to_integer(unsigned(instruct(8 to 11)))))) then
  Reg_bank(to_integer(unsigned(instruct(12 to 15)))) <= std_logic_vector(to_unsigned(1, 16));
else
  Reg_bank(to_integer(unsigned(instruct(12 to 15)))) <=  std_logic_vector(to_unsigned(0, 16));
end if;

DECOP2 <= "0001111"; -- < '7' de XOR
Reg_bank(14) <= Reg_bank(14) + std_logic_vector(to_unsigned(1,16));
-- Transferencia
```

OR

```
when "1010" =>
Reg_bank(to_integer(unsigned(instruct(12 to 15)))) <=
NOT
Reg_bank(to_integer(unsigned(instruct(8 to 11))));
DECOP2 <= "0000010"; -- '10' de NOT
Reg_bank(14) <= Reg_bank(14) +
std_logic_vector(to_unsigned(1,16));
```

PROCESSADOR TRANSFERENCIA

LW

```
when "0000" =>
Reg_bank(to_integer(unsigned(instruct(8
to 11)))) <=
Reg_bank(to_integer(unsigned(instruct(4
to 7))) + to_integer(unsigned(instruct(12 to
15))));
Reg_bank(14) <= Reg_bank(14) +
std_logic_vector(to_unsigned(1,16));
if(to_integer(unsigned(instruct(8 to 11))) =
0 ) then
DECOP2 <= "1110111"; -- '-' de NADA;
output  <= 0;
else
DECOP2 <= "0000001"; -- '0' de Lw;
end if;
```

SW

```
when "0001" =>

Reg_bank(to_integer(unsigned(instruct(4 to 7))) + to_integer(unsigned(instruct(8 to 11)))) <=

Reg_bank(to_integer(unsigned(instruct(12 to 15))));
DECOP2 <= "1001111"; -- '1' de Sw;
Reg_bank(14) <= Reg_bank(14) + std_logic_vector(to_unsigned(1,16));
```

LUI

```
when "1011" =>
Reg_bank(to_integer(unsigned(instruct(4
to 7)))) <=

std_logic_vector(to_unsigned(to_integer(u
nsigned(instruct(8 to 15))), 16));
DECOP2 <= "1100000"; -- 'b' de Lui;
Reg_bank(14) <= Reg_bank(14) +
std_logic_vector(to_unsigned(1,16));
```

PROCESSADOR DESVIO CONDICIONAL

BEQ

```
when "1100" =>
if((Reg_bank(to_integer(unsigned(instruct(
4 to 7))))) =
(Reg_bank(to_integer(unsigned(instruct(8
to 11)))))) then
Reg_bank(14) <=
std_logic_vector(to_unsigned(to_integer(u
nsigned(instruct(12 to 15))),16));
else
Reg_bank(14) <= Reg_bank(14) +
std_logic_vector(to_unsigned(1,16));
end if;
DECOP2 <= "0110001"; -- '12' de BEQ;
```

BLT

```
when "1101" =>
if((Reg_bank(to_integer(unsigned(instruct(
4 to 7))))) <
(Reg_bank(to_integer(unsigned(instruct(8
to 11)))))) then
Reg_bank(14) <=
std_logic_vector(to_unsigned(to_integer(u
nsigned(instruct(12 to 15))),16));
else
Reg_bank(14) <= Reg_bank(14) +
std_logic_vector(to_unsigned(1,16));
end if;
DECOP2 <= "1000010"; -- '13' de BLT;
```

PROCESSADOR DESVIO INCONDICIONAL

J

```
when "1110" => --J OP-1110 GOTO-
000000000000 1110
Reg_bank(14) <=
std_logic_vector(to_unsigned(to_integer(u
nsigned(instruct(4 to 15))),16));
DECOP2 <= "1000010"; -- '13' de BLT;
```

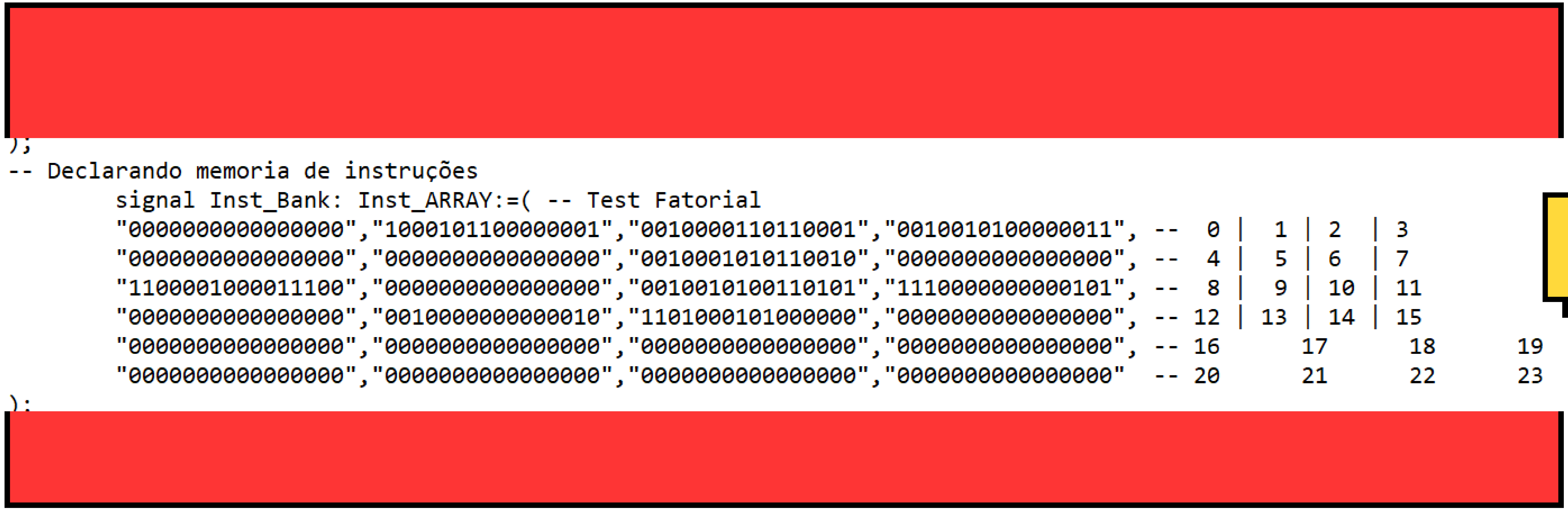
JAL

```
when "1111" => --Jal OP-1111 D-0000
GOTO-00000000
Reg_bank(15) <= Reg_bank(14);
Reg_bank(14) <=
std_logic_vector(to_unsigned(to_integer(u
nsigned(instruct(8 to 15))),16));
DECOP2 <= "1000010"; -- '13' de BLT;
```

OUTROS

```
when others =>
output  <= 0       ; -- '-' de passagem intermediaria
destino <= 0    ; -- '-' de passagem intermediaria
DECOP2  <= "1110111"; -- '-' de passagem intermediaria
```

PROCESSADOR PROGRAMA FATORIAL

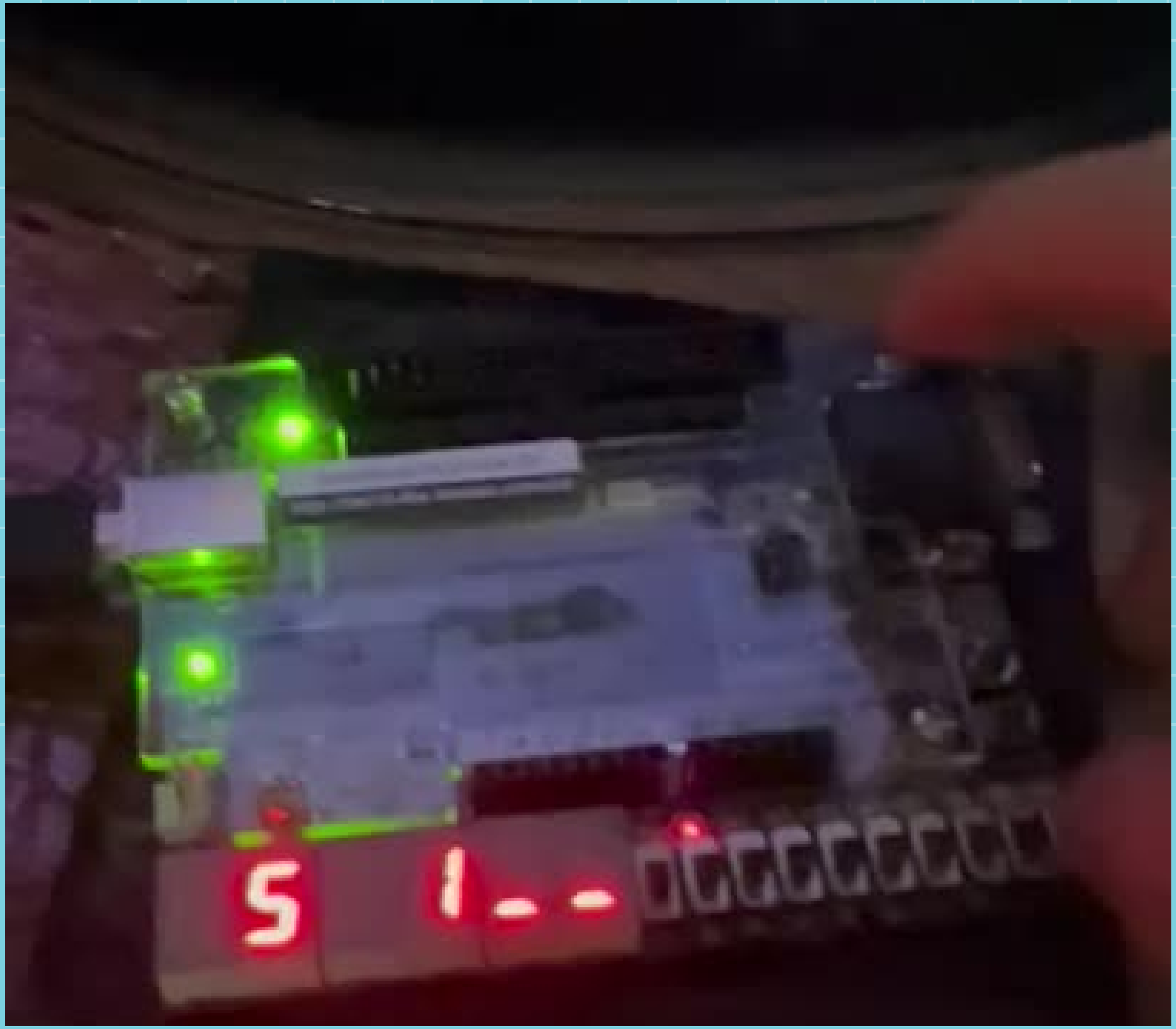


```
[redacted]
);
-- Declarando memoria de instruções
        signal Inst_Bank: Inst_ARRAY:=( -- Test Fatorial
        "0000000000000000","1000101100000001","0010000110110001","0010010100000011", --  0 |  1 | 2  | 3
        "0000000000000000","0000000000000000","0010001010110010","0000000000000000", --  4 |  5 | 6  | 7
        "1100001000011100","0000000000000000","0010010100110101","1110000000000101", --  8 |  9 | 10 | 11
        "0000000000000000","0010000000000010","1101000101000000","0000000000000000", -- 12 | 13 | 14 | 15
        "0000000000000000","0000000000000000","0000000000000000","0000000000000000", -- 16      17      18      19
        "0000000000000000","0000000000000000","0000000000000000","0000000000000000"  -- 20      21      22      23
);
[redacted]
```

Declarando instruções

NUMEROS DA VARIAVEL SAIDA

```
case(output) is
when 0   => DECOP <= "0000001"; --'0'
when 1   => DECOP <= "1001111"; --'1'
when 2   => DECOP <= "0010010"; --'2'
when 3   => DECOP <= "0000110"; --'3'
when 4   => DECOP <= "1001100"; --'4'
when 5   => DECOP <= "0100100"; --'5'
when 6   => DECOP <= "0100000"; --'6'
when 7   => DECOP <= "0001111"; --'7'
when 8  => DECOP <= "0000000"; --'8'
when 9   => DECOP <= "0000100"; --'9'
when others => DECOP <= "1110111"; --'-'
end case;
end if;
led_out <= ciclo and ENABLE;
-- DECOP  <= DEC;
-- DECOP2 <= DEC2;
leds  <= instruct(0 to 3);
leds2 <= instruct(4 to 7);
if(to_integer(unsigned(Reg_bank(14))) = 15) then
DECOP  <= "1110111";
DECOP2 <= "1110111";
end if;
```



☑ processador funcionando

CODIGO