CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS

LABORATÓRIO DE ARQUITETURA E ORGANIZAÇÃO DE COMPUTADORES II

PRÁTICA II - Implementação de um processador

Alunos: ALEXANDRE ROQUE e VITOR SANTANA
Orientadora: Profa. Daniela Cristina Cascini Kupsch

Belo Horizonte

2021

Sumário:

Objetivos e apresentação:

Os objetivos desta prática é a implementação de um processador e de uma TLB no estágio de Instruction Fetch, onde dado um endereço virtual (Contador R7), procurar a tradução para a página física que contém uma instrução e passa-la para o processador.

O processador irá processar e executar a instrução, com base na separação de cada uma delas. O projeto foi feito na linguagem Verilog HDL de descrição de hardware, com auxílio do software Quartus-ModelSim da Altera©.

Separação do endereço e definição das instruções:

Como temos 10 instruções suportadas pelo nosso processador, utilizamos 4 bits do endereço para determinar o código da operação (OpCode), além disso separamos 3 bits para cada um dos registradores “Rx” e “Ry”. Como pode ser visto abaixo: Ry Rx OPCODE → 0000000000000000

Instrução	OpCode	Descrição
LD Rx Ry	0000	Rx ← [[Ry]]
ST Rx Ry	0001	[Ry] ← [Rx
MVNZ Rx Ry	0010	if G != 0, Rx ← [Ry
MV Rx Ry	0011	Rx ← [Ry
MVI Rx #D	0100	Rx ← D
ADD Rx Ry	0101	Rx ← [Rx]+[Ry]
SUB Rx Ry	0110	Rx ← [Rx] − [Ry]
OR Rx Ry	0111	Rx ← [Rx] || [Ry]
SLT Rx Ry	1000	if Rx<Ry, Rx = 1 else Rx = 0
SLL Rx Ry	1001	Rx ← [Rx] << [Ry]
SRL Rx Ry	1010	Rx ← [Rx] >> [Ry]

Vale ressaltar que o imediato é passado separado do endereço, como um dado a mais, são 16 bits de imediato, através do DataIN. O sinal de Imm controla se determinado dado será utilizado ou não, na instrução MVI ele é ativado e em default está sempre em 0.

Estágios temporais:

Operação	T1	T2	T3	T4
LD Rx Ry (0000)	ADDRin, **RY**out	-	LD, **RX**in, DINOut	Done
ST Rx Ry (0001)	DOUTin, **RX**out	**RY**out, W_D, ADDRin	Done	-
MVNZ Rx Ry (0010)	if (G != 0) **RX**in, **RY**out Done	-	-	-
MV Rx Ry (0011)	**RY**out, **RX**in, Done	-	-	-
MVI Rx #D (0100)	Imm, DINout, Done	-	-	-
ADD Rx Ry (0101)	**RX**out, Ain	**RY**out, Gin, ULAOp(0101)	Gout, **RX**in, Done	-
SUB Rx Ry (0110)	**RX**out, Ain	**RY**out, Gin, ULAOp(0110)	Gout, **RX**in, Done	-
OR Rx Ry (0111)	**RX**out, Ain	**RY**out, Gin, ULAOp(0111)	Gout, **RX**in, Done	-
SLT Rx Ry (1000)	**RX**out, Ain	**RY**out, Gin, ULAOp(1000)	Gout, **RX**in, Done	-
SLL Rx Ry (1001)	**RX**out, Ain	**RY**out, Gin, ULAOp(1001)	Gout, **RX**in, Done	-
SRL Rx Ry (1010)	**RX**out, Ain	**RY**out, Gin, ULAOp(1010)	Gout, **RX**in, Done	-

Simulação:

Instruções de teste disponibilizadas no ava:

Instrução	Tradução	R0	R1	R2	R3
MVI R0, #2 Imm	00000000000000000000000000000100 0000000000000010	**2**	0	0	0
MVI R1, #3 Imm	00000000000000000000000000010100 0000000000000011	2	**3**	0	0
ADD R1, R0	00000000000000000000000000010101	2	**5**	0	0
MVI R2, #6 Imm	00000000000000000000000000100100 0000000000000110	2	5	**6**	0
SUB R2, R1	00000000000000000000000010100110	2	5	**1**	0
MV R3, R2	00000000000000000000000100110011	2	5	1	**1**
ADD R0,R3	00000000000000000000000110000101	**3**	5	1	1
OR R1,R0	00000000000000000000000000010111	3	**7**	1	1
SUB R1,R0	00000000000000000000000000010110	3	**4**	1	1
ADD R1, R3	00000000000000000000000110010101	3	**5**	1	1
SLL R1, R3	00000000000000000000000110011001	1	**A**	1	1
SRL R1, R3	00000000000000000000000110011010	1	**5**	1	1
MVI R0, #0 Imm	00000000000000000000000000000100 0000000000000000	**0**	5	1	1
SLT R0, R1	00000000000000000000000010001000	**1**	5	1	1
SLT R1, R1	00000000000000000000000010011000	1	**0**	1	1
MVI R3, #3 Imm	00000000000000000000000000110100 0000000000000011	1	0	1	**3**
MVI R1, #5 Imm	00000000000000000000000000010100 0000000000000101	1	**5**	1	3
ADD R0, R3	00000000000000000000000110000101	**4**	5	1	3
MVI R0, #0 Imm	00000000000000000000000000000100 0000000000000000	**0**	5	1	3
LD R2, R3	00000000000000000000000110100000	0	5	**4**	3
ADD R2, R3	00000000000000000000000110100101	0	5	**7**	3
SD R2, R0	00000000000000000000000000100001	0	5	7	3
LD R0, R0	00000000000000000000000000000000	**7**	5	7	3
SUB R0, R3	00000000000000000000000110000110	**4**	5	7	3
MVI R0, #0 Imm	00000000000000000000000000000100 0000000000000000	**0**	5	7	3
ADD R0, R0	00000000000000000000000000000101	**0**	5	7	3
MVNZ R0, R2	00000000000000000000000100000010	0	5	7	3
SUB R1,R3	00000000000000000000000110010110	0	**2**	7	3
MVNZ R0, R2	00000000000000000000000100000010	**7**	2	7	3
ADD R0, R1	00000000000000000000000010000101	**9**	2	7	3

Código Teste:

Move o valor do Imm 2 para o registrador R0

MVI R0, #2 **Imm**	00000000000000000000000000000100 0000000000000010	**2**	0	0	0

```
enderecoVirtual =     1
DIN = 0000000000000100, DataIN =     2
R0    R1    R2    R3        R7
2     0     0     0         1
STATEMENT 1 :: time is 200
```

Move o valor do Imm 3 para o registrador R1

MVI R1, #3 **Imm**	00000000000000000000000000010100 0000000000000011	2	**3**	0	0

```
enderecoVirtual =     2
DIN = 0000000000010100, DataIN =     3
R0    R1    R2    R3        R7
2     3     0     0         2
STATEMENT 1 :: time is 400
```

Soma do conteúdo de R0 em R1

ADD R1, R0	00000000000000000000000000010101	2	**5**	0	0

```
enderecoVirtual =     3
DIN = 0000000000010101, DataIN =     x
R0    R1    R2    R3        R7
2     5     0     0         3
STATEMENT 1 :: time is 600
```

Move o valor do Imm 6 para o registrador R2

MVI R2, #6 **Imm**	00000000000000000000000000100100 0000000000000110	2	5	**6**	0

```
enderecoVirtual =     4
DIN = 0000000000100100, DataIN =     6
R0    R1    R2    R3        R7
2     5     6     0         4
STATEMENT 1 :: time is 800
```

Subtrai o conteúdo de R2 o valor de R1

SUB R2, R1	00000000000000000000000010100110	2	5	**1**	0

```
enderecoVirtual =     5
DIN = 0000000010100110, DataIN =     x
R0    R1    R2    R3        R7
2     5     1     0         5
STATEMENT 1 :: time is 1000
```

Move o valor do registrador R2 para o registrador R3

MV R3, R2	00000000000000000000000100110011	2	5	1	1

```
enderecoVirtual =     6
DIN = 0000000100110011, DataIN =     x
R0    R1    R2    R3        R7
2     5     1     1         6
STATEMENT 1 :: time is 1200
```

Soma do conteúdo de R3 em R0

ADD R0,R3	00000000000000000000000110000101	3	5	1	1

```
enderecoVirtual =     7
DIN = 0000000110000101, DataIN =     x
R0    R1    R2    R3        R7
3     5     1     1         7
STATEMENT 1 :: time is 1400
```

Operação OR do conteúdo de R0 e R1

OR R1,R0	00000000000000000000000000010111	3	7	1	1

```
enderecoVirtual =     8
DIN = 0000000000010111, DataIN =     x
R0    R1    R2    R3        R7
3     7     1     1         8
STATEMENT 1 :: time is 1600
```

Subtrai o conteúdo de R1 o valor de R0

SUB R1,R0	00000000000000000000000000010110	3	4	1	1

```
enderecoVirtual =     9
DIN = 0000000000010110, DataIN =     x
R0    R1    R2    R3        R7
3     4     1     1         9
STATEMENT 1 :: time is 1800
```

Soma do conteúdo de R3 em R1

ADD R1, R3	00000000000000000000000110010101	3	5	1	1

```
enderecoVirtual =    10
DIN = 0000000110010101, DataIN =     x
R0    R1    R2    R3        R7
3     5     1     1         10
STATEMENT 1 :: time is 2000
```

Operação SLL do conteúdo de R3 e R1

SLL R1, R3	00000000000000000000000110011001	3	A	1	1

```
enderecoVirtual =    11
DIN = 0000000110011001, DataIN =     x
R0    R1    R2    R3        R7
3    10     1     1         11
STATEMENT 1 :: time is 2200
```

Operação SRL do conteúdo de R3 e R1

SRL R1, R3	00000000000000000000000110011010	3	**5**	1	1

```
enderecoVirtual =    12
DIN = 0000000110011010, DataIN =     x
R0    R1    R2    R3       R7
3     5     1     1        12
STATEMENT 1 :: time is 2400
```

Move o valor do Imm 0 para o registrador R0

MVI R0, #0 **Imm**	00000000000000000000000000000100 0000000000000000	**0**	5	1	1

```
enderecoVirtual =    13
DIN = 0000000000000100, DataIN =     0
R0    R1    R2    R3       R7
0     5     1     1        13
STATEMENT 1 :: time is 2600
```

Operação SLT de R1 e R0

SLT R0, R1	00000000000000000000000010001000	**1**	5	1	1

```
enderecoVirtual =    14
DIN = 0000000010001000, DataIN =     x
R0    R1    R2    R3       R7
1     5     1     1        14
STATEMENT 1 :: time is 2800
```

Operação SLT de R1 e R1

SLT R1, R1	00000000000000000000000010011000	1	**0**	1	1

```
enderecoVirtual =    15
DIN = 0000000010011000, DataIN =     x
R0    R1    R2    R3       R7
1     0     1     1        15
STATEMENT 1 :: time is 3000
```

Move o valor do Imm 3 para o registrador R3

MVI R3, #3 **Imm**	00000000000000000000000000110100 0000000000000011	1	0	1	**3**

```
enderecoVirtual =    16
DIN = 0000000000110100, DataIN =     3
R0    R1    R2    R3       R7
1     0     1     3        16
STATEMENT 1 :: time is 3200
```

Move o valor do Imm 5 para o registrador R1

MVI R1, #5 **Imm**	00000000000000000000000000010100 0000000000000101	1	**5**	1	3

```
enderecoVirtual =    17
DIN = 0000000000010100, DataIN =     5
R0     R1     R2     R3        R7
1      5      1      3         17
STATEMENT 1 :: time is 3400
```

Soma do conteúdo de R3 em R0

ADD R0, R3	00000000000000000000000110000101	**4**	5	1	3

```
enderecoVirtual =    18
DIN = 0000000110000101, DataIN =     x
R0     R1     R2     R3        R7
4      5      1      3         18
STATEMENT 1 :: time is 3600
```

Move o valor do Imm 0 para o registrador R0

MVI R0, #0 **Imm**	00000000000000000000000000000100 0000000000000000	**0**	5	1	3

```
enderecoVirtual =    19
DIN = 0000000000000100, DataIN =     0
R0     R1     R2     R3        R7
0      5      1      3         19
STATEMENT 1 :: time is 3800
```

Busca na memória o endereço que está em R3 e armazena em R2

LD R2, R3	00000000000000000000000110100000	0	5	**4**	3

```
enderecoVirtual =    21
DIN = 0000000110100101, DataIN =     x
R0     R1     R2     R3        R7
0      5      4      3         21
STATEMENT 1 :: time is 4200
```

Soma do conteúdo de R3 em R2

ADD R2, R3	00000000000000000000000110100101	0	5	**7**	3

```
enderecoVirtual =    22
DIN = 0000000000100001, DataIN =     x
R0     R1     R2     R3        R7
0      5      7      3         22
STATEMENT 1 :: time is 4400
```

Armazena na memória o conteúdo de R2 no endereço de R0

SD R2, R0	00000000000000000000000000100001	0	5	7	3

```
enderecoVirtual =    23
DIN = 0000000000000000, DataIN    x
R0    R1    R2    R3       R7
0     5     7     3        23
STATEMENT 1 :: time is 4500
```

Busca na memória o endereço que está em R0 e armazena em R0

LD R0, R0	00000000000000000000000000000000	**7**	5	7	3

```
enderecoVirtual =    23
DIN = 0000000000000000, DataIN =     x
R0    R1    R2    R3       R7
7     5     7     3        23
STATEMENT 1 :: time is 4700
```

Subtrai o conteúdo de R0 o valor de R3

SUB R0, R3	00000000000000000000000110000110	**4**	5	7	3

```
enderecoVirtual =    24
DIN = 0000000110000110, DataIN =     x
R0    R1    R2    R3       R7
4     5     7     3        24
STATEMENT 1 :: time is 4900
```

Move o valor do Imm 0 para o registrador R0

MVI R0, #0 **Imm**	00000000000000000000000000000100 0000000000000000	**0**	5	7	3

```
enderecoVirtual =    25
DIN = 0000000000000100, DataIN =     0
R0    R1    R2    R3       R7
0     5     7     3        25
STATEMENT 1 :: time is 5100
```

Soma do conteúdo de R0 em R0

ADD R0, R0	00000000000000000000000000000101	**0**	5	7	3

```
enderecoVirtual =    26
DIN = 0000000000000101, DataIN =     x
R0    R1    R2    R3       R7
0     5     7     3        26
STATEMENT 1 :: time is 5300
```

Se G for diferente 0, atribui o valor de R2 em R0

MVNZ R0, R2	00000000000000000000000100000010	0	5	7	3

```
enderecoVirtual =    27
DIN = 0000000100000010, DataIN =     x
R0     R1     R2     R3        R7
0      5      7      3         27
STATEMENT 1 :: time is 5500
```

Subtrai o conteúdo de R1 o valor de R3

SUB R1,R3	00000000000000000000000110010110	0	**2**	7	3

```
enderecoVirtual =    28
DIN = 0000000110010110, DataIN =     x
R0     R1     R2     R3        R7
0      2      7      3         28
STATEMENT 1 :: time is 5700
```

Se G for diferente 0, atribui o valor de R2 em R0

MVNZ R0, R2	00000000000000000000000100000010	**7**	2	7	3

```
enderecoVirtual =    29
DIN = 0000000100000010, DataIN =     x
R0     R1     R2     R3        R7
7      2      7      3         29
STATEMENT 1 :: time is 5900
```

Soma do conteúdo de R1 em R0

ADD R0, R1	00000000000000000000000010000101	**9**	2	7	3

```
enderecoVirtual =    30
DIN = 0000000010000101, DataIN =     x
R0     R1     R2     R3        R7
9      2      7      3         30
STATEMENT 1 :: time is 6100
```

Instruções para o Loop:

Instrução	Tradução	R2	R4	R5	R7
MVI R2, #1 **Imediato**	00000000000000000000000000100100 0000000000000001	**1**	0	0	1
MVI R4, #10 **Imediato**	00000000000000000000000001000100 0000000000001010	1	**10**	0	2
MV R5,R7	00000000000000000000001111010011	1		**3**	3
SUB R4, R2	00000000000000000000000101000110	1	10...0	3	4
MVNZ R7,R5	00000000000000000000001011110010	1		3	**3**

Loop:

Move o valor do Imm 1 para o registrador R2

MVI R2, #1 **Imediato**	00000000000000000000000000100100 0000000000000001	**1**	0	0	1

```
enderecoVirtual =     1
DIN = 0000000000100100, DataIN =     1
R2    R4    R5    R7
1     0     0     1
STATEMENT 1 :: time is 200
```

Move o valor do Imm 10 para o registrador R4

MVI R4, #10 **Imediato**	00000000000000000000000001000100 0000000000001010	1	**10**	0	2

```
enderecoVirtual =     2
DIN = 0000000001000100, DataIN =    10
R2    R4    R5    R7
1    10     0     2
STATEMENT 1 :: time is 400
```

Move o conteúdo do registrador R7 para o registrador R5

MV R5,R7	00000000000000000000001111010011	1	10	**3**	3

```
enderecoVirtual =     3
DIN = 0000001111010011, DataIN =     x
R2    R4    R5    R7
1    10     3     3
STATEMENT 1 :: time is 600
```

Subtrai o conteúdo de R4 o valor de R2

SUB R4, R2	00000000000000000000000101000110	1	**9**	3	4

```
enderecoVirtual =     4
DIN = 0000000101000110, DataIN =     x
R2    R4    R5    R7
1     9     3     4
STATEMENT 1 :: time is 800
```

Se G for diferente 0, atribui o valor de R5 em R7

MVNZ R7,R5	00000000000000000000001011110010	1	9	3	**3**

```
enderecoVirtual =     3
DIN = 0000001111010011, DataIN =     x
R2    R4    R5    R7
1     9     3     3
STATEMENT 1 :: time is 1000
```

Subtrai o conteúdo de R4 o valor de R2

SUB R4, R2	00000000000000000000000101000110	1	**8**	3	4

```
enderecoVirtual =     4
DIN = 0000000101000110, DataIN =     x
R2    R4    R5    R7
1     8     3     4
STATEMENT 1 :: time is 1200
```

Se G for diferente 0, atribui o valor de R5 em R7

MVNZ R7,R5	00000000000000000000001011110010	1	8	3	**3**

```
enderecoVirtual =     3
DIN = 0000001111010011, DataIN =     x
R2    R4    R5    R7
1     8     3     3
STATEMENT 1 :: time is 1400
```

Subtrai o conteúdo de R4 o valor de R2

SUB R4, R2	00000000000000000000000101000110	1	**7**	3	4

```
enderecoVirtual =     4
DIN = 0000000101000110, DataIN =     x
R2    R4    R5    R7
1     7     3     4
STATEMENT 1 :: time is 1600
```

Se G for diferente 0, atribui o valor de R5 em R7

MVNZ R7,R5	00000000000000000000001011110010	1	7	3	**3**

```
enderecoVirtual =     3
DIN = 0000001111010011, DataIN =     x
R2    R4    R5    R7
1     7     3     3
STATEMENT 1 :: time is 1800
```

Subtrai o conteúdo de R4 o valor de R2

SUB R4, R2	00000000000000000000000101000110	1	**6**	3	4

```
enderecoVirtual =     4
DIN = 0000000101000110, DataIN =     x
R2    R4    R5    R7
1     6     3     4
STATEMENT 1 :: time is 2000
```

Se G for diferente 0, atribui o valor de R5 em R7

MVNZ R7,R5	00000000000000000000001011110010	1	6	3	**3**

```
enderecoVirtual =     3
DIN = 0000001111010011, DataIN =     x
R2    R4    R5    R7
1     6     3     3
STATEMENT 1 :: time is 2200
```

Subtrai o conteúdo de R4 o valor de R2

SUB R4, R2	00000000000000000000000101000110	1	**5**	3	4

```
enderecoVirtual =     4
DIN = 0000000101000110, DataIN =     x
R2    R4    R5    R7
1     5     3     4
STATEMENT 1 :: time is 2400
```

Se G for diferente 0, atribui o valor de R5 em R7

MVNZ R7,R5	00000000000000000000001011110010	1	5	3	**3**

```
enderecoVirtual =     3
DIN = 0000001111010011, DataIN =     x
R2    R4    R5    R7
1     5     3     3
STATEMENT 1 :: time is 2600
```

Subtrai o conteúdo de R4 o valor de R2

SUB R4, R2	00000000000000000000000101000110	1	**4**	3	4

```
enderecoVirtual =     4
DIN = 0000000101000110, DataIN =     x
R2    R4    R5    R7
1     4     3     4
STATEMENT 1 :: time is 2800
```

Se G for diferente 0, atribui o valor de R5 em R7

MVNZ R7,R5	00000000000000000000001011110010	1	4	3	**3**

```
enderecoVirtual =     3
DIN = 0000001111010011, DataIN =     x
R2    R4    R5    R7
1     4     3     3
STATEMENT 1 :: time is 3000
```

Subtrai o conteúdo de R4 o valor de R2

SUB R4, R2	00000000000000000000000101000110	1	**3**	3	4

```
enderecoVirtual =     4
DIN = 0000000101000110, DataIN =     x
R2    R4    R5    R7
1     3     3     4
STATEMENT 1 :: time is 3200
```

Se G for diferente 0, atribui o valor de R5 em R7

MVNZ R7,R5	00000000000000000000001011110010	1	3	3	**3**

```
enderecoVirtual =     3
DIN = 0000001111010011, DataIN =     x
R2    R4    R5    R7
1     3     3     3
STATEMENT 1 :: time is 3400
```

Subtrai o conteúdo de R4 o valor de R2

SUB R4, R2	00000000000000000000000101000110	1	**2**	3	4

```
enderecoVirtual =     4
DIN = 0000000101000110, DataIN =     x
R2    R4    R5    R7
1     2     3     4
STATEMENT 1 :: time is 3600
```

Se G for diferente 0, atribui o valor de R5 em R7

MVNZ R7,R5	00000000000000000000001011110010	1	2	3	**3**

```
enderecoVirtual =     3
DIN = 0000001111010011, DataIN =     x
R2    R4    R5    R7
1     2     3     3
STATEMENT 1 :: time is 3800
```

Subtrai o conteúdo de R4 o valor de R2

SUB R4, R2	00000000000000000000000101000110	1	**1**	3	4

```
enderecoVirtual =     4
DIN = 0000000101000110, DataIN =     x
R2    R4    R5    R7
1     1     3     4
STATEMENT 1 :: time is 4000
```

Se G for diferente 0, atribui o valor de R5 em R7

MVNZ R7,R5	00000000000000000000001011110010	1	1	3	**3**

```
enderecoVirtual =     3
DIN = 0000001111010011, DataIN =     x
R2    R4    R5    R7
1     1     3     3
STATEMENT 1 :: time is 4200
```

Subtrai o conteúdo de R4 o valor de R2

SUB R4, R2	00000000000000000000000101000110	1	**0**	3	4

```
enderecoVirtual =     4
DIN = 0000000101000110, DataIN =     x
R2    R4    R5    R7
1     0     3     4
STATEMENT 1 :: time is 4400
```

Se G diferente 0, atribui o valor de R5 em R7. Como é igual a 0, R7 começa a contar, como pode ser visto nas figuras a seguir.

MVNZ R7,R5	00000000000000000000001011110010	1	6	3	**5**

```
enderecoVirtual =     5
DIN = 0000001011110010, DataIN    x
R2    R4    R5    R7
1     0     3     5
STATEMENT 1 :: time is 4500

enderecoVirtual =     6
DIN = xxxxxxxxxxxxxxxx, DataIN =     x
R2    R4    R5    R7
1     0     3     6
STATEMENT 1 :: time is 4700
```

Mudanças:

Devido aos comentários da professora sobre o TLB após a apresentação, foram feitas as seguintes alterações:

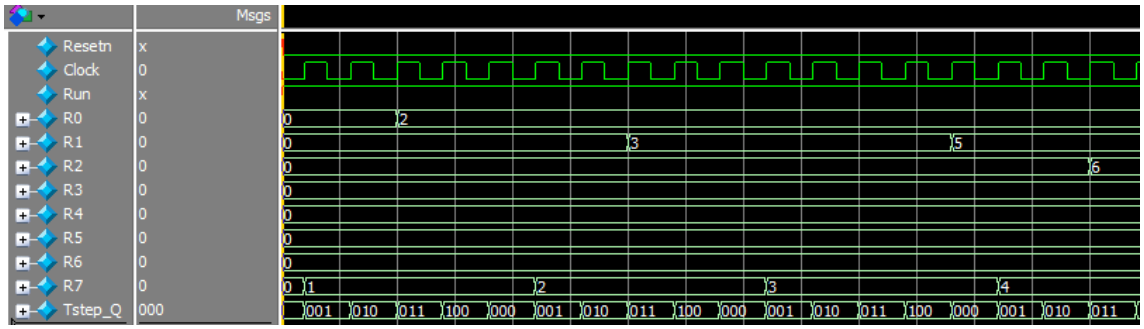
Foram adicionados mais bits para a página virtual, visto que essa passa a sensação para o programador de uma memória maior que realmente existe.

Como mencionado pela professora deixamos o TLB diretamente mapeada, para resolver tal problema acrescentamos um laço de repetição para percorrer todas as posições do TLB, com isso ela fica conforme solicitado “totalmente associativo”.

```
 7  module TLB(enderecoVirtual, Clock, DIN, Data);
 8     input [15:0] enderecoVirtual;
 9     input Clock;
10     output reg [15:0] DIN;
11     output reg [15:0] Data;
12
13     reg dirty [50:0];
14     reg valido [50:0];
15
16     reg [31:0] PaginaVirtual [50:0];
17
18     reg [15:0] Dados         [50:0];
```

```
for(i = 0; i < 50; i = i + 1)begin
   if(i == Indice)begin
      if(valido[Indice] == 1)begin
         Hit = 1;
         DIN = PaginaVirtual[Indice][15:0];
         Data= Dados[Indice];
      end
   end
end
```

Outra mudança feita foi a instanciação de um TopLevel que tem como saída os registradores R0,...,R7 e o passo TStep_Q. Para facilitar a visualização do projeto e das suas execuções na simulação do ModelSim.



Conclusões:

A prática ajudou os integrantes do grupo a relembrarem os conceitos estudados na disciplina de Arquitetura e Organização de Computadores 1 e consolidou conhecimentos sobre processadores e a sua comunicação com as memórias implementadas, ajudando também a refletir sobre o componente TLB (Translation Lookaside Buffer).