



UNIVERSIDADE FEDERAL DE CAMPINA GRANDE  
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA  
UNIDADE ACADÊMICA DE SISTEMAS E COMPUTAÇÃO  
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

**THAYANE STHEFFANY SILVA BARROS**

**ROTEIRO 9**

# Problema 1 –Memória de Instruções

## 3) a)Qual é o endereço de memória que aponta para a primeira instrução?

O endereço de memória que aponta para a primeira instrução é 0x0.

The screenshot shows the Chocopy simulator interface. The 'PC' column in the instruction list is highlighted with a red arrow pointing to the first instruction at address 0x0. The instruction is 'jal x0 76' with machine code 0x04C0006F. The right panel shows the registers, with the 'PC' register (labeled as 'PC' in the original image, but 'PC' is not explicitly labeled in the registers list, it's the first entry) containing the value 0x0.

PC	Machine Code	Basic Code	Original Code
0x0	0x04C0006F	jal x0 76	j main
0x4	0xFF010113	addi x2 x2 -8	addi sp, sp, -8
0x8	0x00C12023	sw x12 0(x2)	sw a2, 0(sp)
0xc	0x00112223	sw x1 4(x2)	sw ra, 4(sp)
0x10	0x00200293	addi x5 x0 2	addi t0, x0, 2
0x14	0x005622B3	slt x5 x12 x5	slt t0, a2, t0
0x18	0x00028C63	beq x5 x0 24	beq t0, x0, anotherCall
0x1c	0x00412083	lw x1 4(x2)	lw ra, 4(sp)
0x20	0x00012603	lw x12 0(x2)	lw a2, 0(sp)
0x24	0x00010113	addi x2 x2 8	addi sp, sp, 8
0x28	0x00100513	addi x10 x0 1	addi a0, x0, 1
0x2c	0x00000067	jair x0 x1 0	jr ra
0x30	0xFFFF0013	addi x12 x12 -1	addi a2, a2, -1

## 3) b)Qual é o endereço de memória que aponta para a última instrução?

O endereço de memória que aponta para a última instrução é 0x50.

The screenshot shows the Chocopy simulator interface. The 'PC' column in the instruction list is highlighted with a red arrow pointing to the last instruction at address 0x50. The instruction is 'jal x1 -76' with machine code 0xFB5FF0EF. The right panel shows the registers, with the 'PC' register (labeled as 'PC' in the original image, but 'PC' is not explicitly labeled in the registers list, it's the first entry) containing the value 0x50.

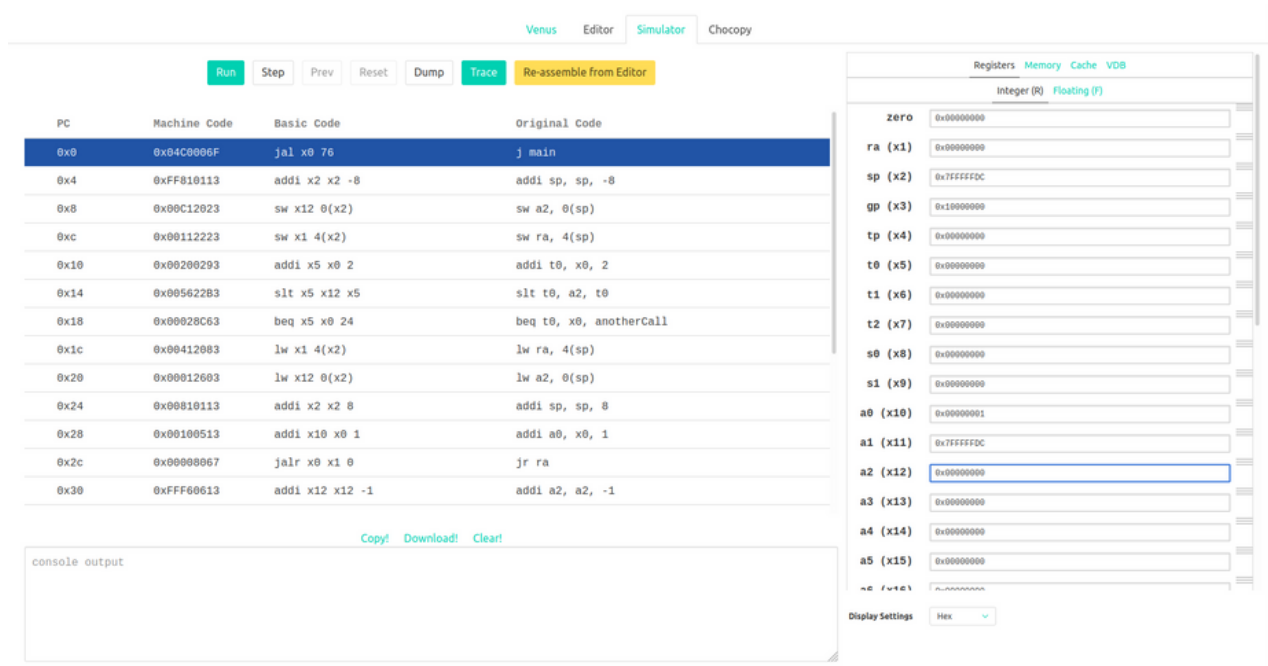
PC	Machine Code	Basic Code	Original Code
0x1c	0x00412083	lw x1 4(x2)	lw ra, 4(sp)
0x20	0x00012603	lw x12 0(x2)	lw a2, 0(sp)
0x24	0x00010113	addi x2 x2 8	addi sp, sp, 8
0x28	0x00100513	addi x10 x0 1	addi a0, x0, 1
0x2c	0x00000067	jair x0 x1 0	jr ra
0x30	0xFFFF0013	addi x12 x12 -1	addi a2, a2, -1
0x34	0xFD1FF0EF	jal x1 -48	jal factorialRec
0x38	0x00412083	lw x1 4(x2)	lw ra, 4(sp)
0x3c	0x00012603	lw x12 0(x2)	lw a2, 0(sp)
0x40	0x00010113	addi x2 x2 8	addi sp, sp, 8
0x44	0x02A00533	mul x10 x12 x10	mul a0, a2, a0
0x48	0x00000067	jair x0 x1 0	jr ra
0x4c	0x00500013	addi x12 x0 5	addi a2, x0, 5
0x50	0xFB5FF0EF	jal x1 -76	jal factorialRec

## 3)c) Qual é o espaço de memória ocupado pelo programa (lembrete: cada endereço aponta para 1 byte)?

O espaço de memória ocupado pelo programa é de 84 bytes. Isso pode ser calculado multiplicando o número de instruções no programa pelo número de bytes que cada instrução ocupa. Nesse caso, temos 21 instruções e cada instrução ocupa 4 bytes:  $21 \text{ instruções} * 4 \text{ bytes/instrução} = 84 \text{ bytes}$ .

#### 4.a)Qual é o conteúdo do registrador que armazena valor de “n”?

O registrador que armazena o valor de n é o a2 e inicialmente o seu conteúdo é 0x00000000. Isso indica que o valor de "n" é zero.

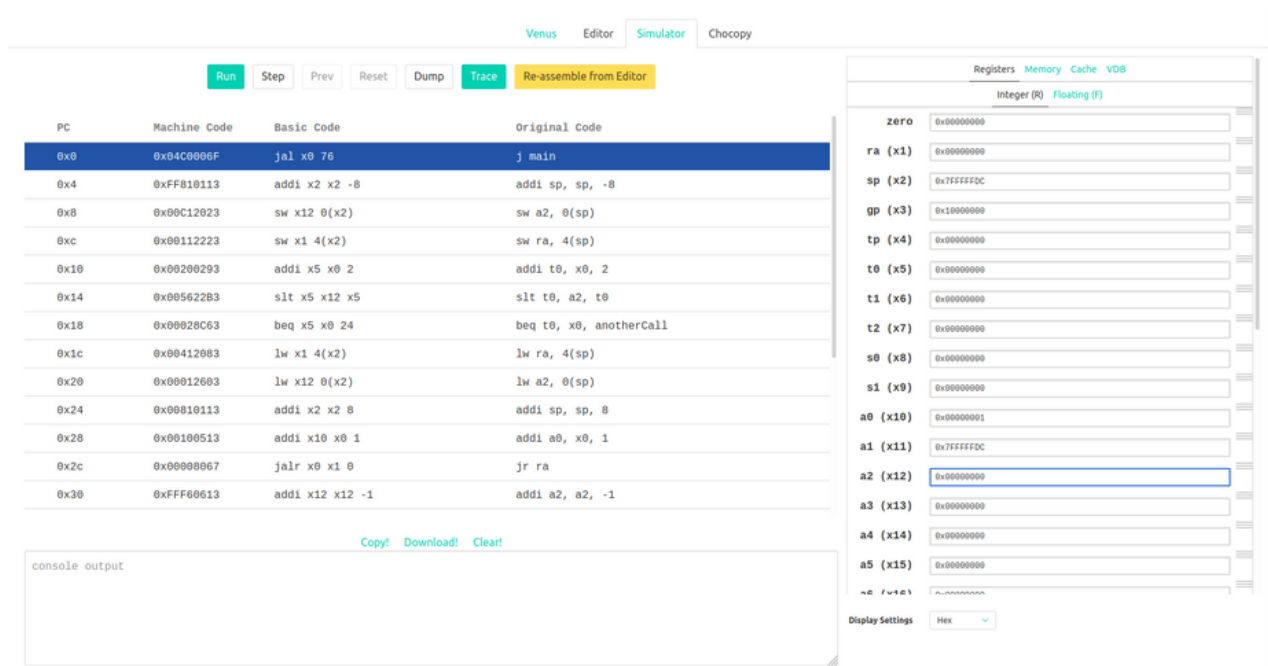


The screenshot shows the Chocopy simulator interface. The 'Registers' panel on the right displays the initial values of 16 registers. Register 'a2 (x12)' is highlighted with a red arrow, showing a value of 0x00000000. The 'PC' panel on the left shows the initial instruction at address 0x0: 'jal x0 76' (j main).

PC	Machine Code	Basic Code	Original Code
0x0	0x04C0006F	jal x0 76	j main
0x4	0xFF810113	addi x2 x2 -8	addi sp, sp, -8
0x8	0x00C12023	sw x12 0(x2)	sw a2, 0(sp)
0xc	0x00112223	sw x1 4(x2)	sw ra, 4(sp)
0x10	0x00200293	addi x5 x0 2	addi t0, x0, 2
0x14	0x005622B3	slt x5 x12 x5	slt t0, a2, t0
0x18	0x00028C63	beq x5 x0 24	beq t0, x0, anotherCall
0x1c	0x00412083	lw x1 4(x2)	lw ra, 4(sp)
0x20	0x00012603	lw x12 0(x2)	lw a2, 0(sp)
0x24	0x00810113	addi x2 x2 8	addi sp, sp, 8
0x28	0x00100513	addi x10 x0 1	addi a0, x0, 1
0x2c	0x00000067	jalr x0 x1 0	jr ra
0x30	0xFFFF00613	addi x12 x12 -1	addi a2, a2, -1

#### 4.b)Qual é o conteúdo do registrador que armazena o valor de “factorial(n)”?

O registrador que armazena o valor de fatorial é o a0 e inicialmente o seu conteúdo é 0x00000001. Isso indica que o valor de fatorial é inicializado com 1.



The screenshot shows the Chocopy simulator interface. The 'Registers' panel on the right displays the initial values of 16 registers. Register 'a0 (x10)' is highlighted with a red arrow, showing a value of 0x00000001. The 'PC' panel on the left shows the initial instruction at address 0x0: 'jal x0 76' (j main).

PC	Machine Code	Basic Code	Original Code
0x0	0x04C0006F	jal x0 76	j main
0x4	0xFF810113	addi x2 x2 -8	addi sp, sp, -8
0x8	0x00C12023	sw x12 0(x2)	sw a2, 0(sp)
0xc	0x00112223	sw x1 4(x2)	sw ra, 4(sp)
0x10	0x00200293	addi x5 x0 2	addi t0, x0, 2
0x14	0x005622B3	slt x5 x12 x5	slt t0, a2, t0
0x18	0x00028C63	beq x5 x0 24	beq t0, x0, anotherCall
0x1c	0x00412083	lw x1 4(x2)	lw ra, 4(sp)
0x20	0x00012603	lw x12 0(x2)	lw a2, 0(sp)
0x24	0x00810113	addi x2 x2 8	addi sp, sp, 8
0x28	0x00100513	addi x10 x0 1	addi a0, x0, 1
0x2c	0x00000067	jalr x0 x1 0	jr ra
0x30	0xFFFF00613	addi x12 x12 -1	addi a2, a2, -1

### 5.a)Qual é o conteúdo do registrador que armazena o valor de “n”?

Após rodar o código o conteúdo do registrador que armazena o valor de n (a2) é 0x00000005. Isso indica que o valor de "n" é 5.

The screenshot shows the Chocopy simulator interface. On the left, a table displays assembly code with columns for PC, Machine Code, Basic Code, and Original Code. The code includes instructions like `jal x0 76`, `addi x2 x2 -8`, `sw x12 0(x2)`, `sw x1 4(x2)`, `addi x5 x0 2`, `slt x5 x12 x5`, `beq x5 x0 24`, `lw x1 4(x2)`, `lw x12 0(x2)`, `addi x2 x2 8`, `addi x10 x0 1`, `jalr x0 x1 0`, and `addi x12 x12 -1`. Below the table is a console output area. On the right, the 'Registers' panel shows the state of various registers. The register `a2 (x12)` is highlighted with a blue border and contains the value `0x00000005`. A red arrow points to this register. The 'Display Settings' are set to 'Hex'.

PC	Machine Code	Basic Code	Original Code
0x0	0x04C0006F	<code>jal x0 76</code>	<code>j main</code>
0x4	0xFF810113	<code>addi x2 x2 -8</code>	<code>addi sp, sp, -8</code>
0x8	0x00C12023	<code>sw x12 0(x2)</code>	<code>sw a2, 0(sp)</code>
0xc	0x00112223	<code>sw x1 4(x2)</code>	<code>sw ra, 4(sp)</code>
0x10	0x00200293	<code>addi x5 x0 2</code>	<code>addi t0, x0, 2</code>
0x14	0x005622B3	<code>slt x5 x12 x5</code>	<code>slt t0, a2, t0</code>
0x18	0x00028C63	<code>beq x5 x0 24</code>	<code>beq t0, x0, anotherCall</code>
0x1c	0x00412083	<code>lw x1 4(x2)</code>	<code>lw ra, 4(sp)</code>
0x20	0x00012603	<code>lw x12 0(x2)</code>	<code>lw a2, 0(sp)</code>
0x24	0x00010113	<code>addi x2 x2 8</code>	<code>addi sp, sp, 8</code>
0x28	0x00100513	<code>addi x10 x0 1</code>	<code>addi a0, x0, 1</code>
0x2c	0x00000067	<code>jalr x0 x1 0</code>	<code>jr ra</code>
0x30	0xFFFF0013	<code>addi x12 x12 -1</code>	<code>addi a2, a2, -1</code>

### 5.b)Qual é o conteúdo do registrador que armazena o valor de “factorial(n)”?

Após rodar o código o conteúdo do registrador que armazena o valor de fatorial (a0) é 0x00000078 (120 em decimal)

This screenshot is identical to the one above, showing the same assembly code and register state. The register `a0 (x10)` is highlighted with a blue border and contains the value `0x00000078`. A red arrow points to this register, indicating the result of the factorial calculation.

# Problema 2 –Memória Cache

2)Selecionar “Run” (desconsiderar qualquer mensagem de erro exibida) e informar os valores das variáveis a seguir.

Hit Count: 2

Accesses: 20

Hit Rate: 0.1

VenusEditorSimulatorChocopy

RunStepPrevResetDumpTraceRe-assemble from Editor

PC	Machine Code	Basic Code	Original Code
0x0	0x04C0006F	j al x0 76	j main
0x4	0xFF010113	addi x2 x2 -8	addi sp, sp, -8
0x8	0x00C12023	sw x12 0(x2)	sw a2, 0(sp)
0xc	0x00112223	sw x1 4(x2)	sw ra, 4(sp)
0x10	0x00200293	addi x5 x0 2	addi t0, x0, 2
0x14	0x005622B3	slt x5 x12 x5	slt t0, a2, t0
0x18	0x00028C63	beq x5 x0 24	beq t0, x0, anotherCall
0x1c	0x00412083	lw x1 4(x2)	lw ra, 4(sp)
0x20	0x00012603	lw x12 0(x2)	lw a2, 0(sp)
0x24	0x00010113	addi x2 x2 8	addi sp, sp, 8
0x28	0x00100513	addi x10 x0 1	addi a0, x0, 1
0x2c	0x00000067	j alr x0 x1 0	j r ra
0x30	0xFFFF00613	addi x12 x12 -1	addi a2, a2, -1

Copy!Download!Clear!

console output

RegistersMemoryCacheVDB

Cache Levels1

Block Size (Bytes)4

Number of Blocks2

Associativity1

Cache Size (Bytes)8

Enable?Enables current selected level of the cache.

Direct Mapped

LRU L1

Hit Count2

Accesses20

Hit Rate0.1

01 MISS01 MISS

NOTE: This is a write through, write allocate cache.

Seed6779040055319010227

Display SettingsHex

**5) Selecionar “Run” (desconsiderar qualquer mensagem de erro exibida) e informar os valores das variáveis a seguir. Comparando com o resultado anterior, o que é possível concluir?**

Hit Count: 10

Accesses: 20

Hit Rate: 0.5

The screenshot shows the Venus simulator interface. The main window displays assembly code with columns for PC, Machine Code, Basic Code, and Original Code. The code includes instructions like `jal x0 76`, `addi x2 x2 -8`, `sw x12 0(x2)`, `sw x1 4(x2)`, `addi x5 x0 2`, `slt x5 x12 x5`, `beq x5 x0 24`, `lw x1 4(x2)`, `lw x12 0(x2)`, `addi x2 x2 8`, `addi x10 x0 1`, `jalr x0 x1 0`, and `addi x12 x12 -1`. The right sidebar shows cache settings: Cache Levels (1), Block Size (8 bytes), Number of Blocks (2), Associativity (1), Cache Size (16 bytes), and an 'Enable?' button. Below these, it shows 'Direct Mapped' mapping, LRU and L1 policies, and the results: Hit Count (10), Accesses (20), and Hit Rate (0.5). A bar chart shows 10 hits (green) and 10 misses (red). A note states: 'NOTE: This is a write through, write allocate cache.' The bottom left shows a console output area.

PC	Machine Code	Basic Code	Original Code
0x0	0x04C0006F	jal x0 76	j main
0x4	0xFF810113	addi x2 x2 -8	addi sp, sp, -8
0x8	0x00C12023	sw x12 0(x2)	sw a2, 0(sp)
0xc	0x00112223	sw x1 4(x2)	sw ra, 4(sp)
0x10	0x00200293	addi x5 x0 2	addi t0, x0, 2
0x14	0x005622B3	slt x5 x12 x5	slt t0, a2, t0
0x18	0x00028C63	beq x5 x0 24	beq t0, x0, anotherCall
0x1c	0x00412083	lw x1 4(x2)	lw ra, 4(sp)
0x20	0x00012603	lw x12 0(x2)	lw a2, 0(sp)
0x24	0x00010113	addi x2 x2 8	addi sp, sp, 8
0x28	0x00100513	addi x10 x0 1	addi a0, x0, 1
0x2c	0x00000067	jalr x0 x1 0	jr ra
0x30	0xFFFF00613	addi x12 x12 -1	addi a2, a2, -1

Comparando os resultados das duas configurações de cache, podemos concluir que o aumento no tamanho do bloco (block size) resulta em um melhor desempenho em termos de taxa de acertos (hit rate).

Na primeira configuração, com um tamanho de bloco de 4 bytes e 2 blocos, obtivemos um hit rate de 0.1, o que significa que apenas 10% das acessos foram bem-sucedidos. Já na segunda configuração, com um tamanho de bloco de 8 bytes e 2 blocos, obtivemos um hit rate de 0.5, o que significa que metade das acessos foram bem-sucedidas.

Isso ocorre porque um tamanho de bloco maior permite armazenar mais instruções ou dados em cada bloco da cache. Com mais dados armazenados na cache, há uma maior probabilidade de que as acessos futuras encontrem os dados desejados já na cache, resultando em uma taxa de acertos maior.

Portanto, concluímos que aumentar o tamanho do bloco na cache pode melhorar o desempenho geral do sistema, reduzindo o número de acessos à memória principal.

8)Selecionar “Run” (desconsiderar qualquer mensagem de erro exibida) e informar os valores das variáveis a seguir. Comparando com o resultado anterior, o que é possível concluir?

Hit Count: 14

Accesses: 20

Hit Rate: 0.7

The screenshot shows the Venus MIPS simulator interface. The main window displays a table of instructions being executed. The right sidebar shows the Cache configuration and execution statistics.

PC	Machine Code	Basic Code	Original Code
0x0	0x04C0006F	j al x0 76	j main
0x4	0xFF810113	addi x2 x2 -8	addi sp, sp, -8
0x8	0x00C12023	sw x12 0(x2)	sw a2, 0(sp)
0xc	0x00112223	sw x1 4(x2)	sw ra, 4(sp)
0x10	0x00200293	addi x5 x0 2	addi t0, x0, 2
0x14	0x005622B3	slt x5 x12 x5	slt t0, a2, t0
0x18	0x00020C63	beq x5 x0 24	beq t0, x0, anotherCall
0x1c	0x00412003	lw x1 4(x2)	lw ra, 4(sp)
0x20	0x00012003	lw x12 0(x2)	lw a2, 0(sp)
0x24	0x00010113	addi x2 x2 8	addi sp, sp, 8
0x28	0x00100513	addi x10 x0 1	addi a0, x0, 1
0x2c	0x00000067	jalr x0 x1 0	jr ra
0x30	0xFFFF0013	addi x12 x12 -1	addi a2, a2, -1

Cache configuration (right sidebar):

- Cache Levels: 1
- Block Size (Bytes): 8
- Number of Blocks: 8
- Associativity: 1
- Cache Size (Bytes): 64
- Enable?: ☒ Enables current selected level of the cache.
- Direct Mapped: ☒
- LRU: ☒ L1
- Hit Count: 14
- Accesses: 20
- Hit Rate: 0.7

NOTE: This is a write through, write allocate cache.

Seed: 6779040055319010227

Display Settings: Hex

Comparando o novo resultado com a configuração anterior, podemos concluir que o aumento no número de blocos na cache também contribui para um melhor desempenho em termos de taxa de acertos (hit rate).

Na configuração anterior, com um número de blocos de 2, obtivemos um hit rate de 0.5, ou seja, metade das acessos foram bem-sucedidas.

Porém, na nova configuração, com um número de blocos aumentado para 8, obtivemos um hit rate de 0.7, o que significa que 70% das acessos foram bem-sucedidas.

Isso ocorre porque um maior número de blocos na cache aumenta a capacidade de armazenamento da cache. Com mais blocos disponíveis, há uma maior probabilidade de que os dados e instruções necessários estejam presentes na cache, reduzindo a necessidade de acessar a memória principal.

Portanto, concluímos que tanto o aumento no tamanho do bloco quanto o aumento no número de blocos contribuem para um melhor desempenho da cache e uma maior taxa de acertos.

**9)(ATIVIDADE ADICIONAL) Realizar uma análise comparativa do resultado obtido no item anterior (Mapeamento Direto) coma estratégia de Mapeamento Associativo.**

Utilizando essas configurações:

- Cache Levels = 1
- Block Size (Bytes) = 4
- Number of Blocks = 2

Obtemos:

- Hit Count: 2
- Accesses: 20
- Hit Rate: 0.1

The screenshot displays the Venus MIPS simulator interface. The main window shows assembly code with columns for PC, Machine Code, Basic Code, and Original Code. The code includes instructions like `jal x0 76`, `addi x2 x2 -8`, `sw x12 0(x2)`, `sw x1 4(x2)`, `addi x5 x0 2`, `slt x5 x12 x5`, `beq x5 x0 24`, `lw x1 4(x2)`, `lw x12 0(x2)`, `addi x2 x2 8`, `addi x10 x0 1`, `jalr x0 x1 0`, and `addi x12 x12 -1`.

On the right side, the 'Cache' settings are configured as follows:

- Cache Levels: 1
- Block Size (Bytes): 4
- Number of Blocks: 2
- Associativity: 2
- Cache Size (Bytes): 8
- Enable? (checked)
- Fully Associative (selected)
- LRU (selected)
- L1 (selected)
- Hit Count: 2
- Accesses: 20
- Hit Rate: 0.1

Below the settings, a bar chart shows 0 hits (red) and 20 misses (green). A note states: "NOTE: This is a write through, write allocate cache." The seed is 6779048055319018227.

At the bottom, there is a 'console output' area and 'Display Settings' set to 'Hex'.

Utilizando essas configurações:

- Cache Levels = 1
- Block Size (Bytes) = 8
- Number of Blocks = 2

Obtemos:

- Hit Count: 2
- Accesses: 20
- Hit Rate: 0.1



Venus Editor Simulator Chocopy

Run Step Prev Reset Dump Trace Re-assemble from Editor

PC	Machine Code	Basic Code	Original Code
0x0	0x04C0006F	jal x0 76	j main
0x4	0xFF810113	addi x2 x2 -8	addi sp, sp, -8
0x8	0x00C12023	sw x12 0(x2)	sw a2, 0(sp)
0xc	0x00112223	sw x1 4(x2)	sw ra, 4(sp)
0x10	0x00200293	addi x5 x0 2	addi t0, x0, 2
0x14	0x005622B3	slt x5 x12 x5	slt t0, a2, t0
0x18	0x00028C63	beq x5 x0 24	beq t0, x0, anotherCall
0x1c	0x00412083	lw x1 4(x2)	lw ra, 4(sp)
0x20	0x00012603	lw x12 0(x2)	lw a2, 0(sp)
0x24	0x00810113	addi x2 x2 8	addi sp, sp, 8
0x28	0x00100513	addi x10 x0 1	addi a0, x0, 1
0x2c	0x00000067	jalr x0 x1 0	jr ra
0x30	0xFFFF00613	addi x12 x12 -1	addi a2, a2, -1

console output

Copy! Download! Clear!

Registers Memory **Cache** VDB

Cache Levels: 1

Block Size (Bytes): 8

Number of Blocks: 2

Associativity: 2

Cache Size (Bytes): 16

Enable? Enables current selected level of the cache.

Fully Associative

LRU L1

Hit Count: 2

Accesses: 20

Hit Rate: 0.1

0) MISS

1) MISS

NOTE: This is a write through, write allocate cache.

Seed: 6779040655319010227

Display Settings: Hex

Utilizando essas configurações:

- Cache Levels = 1
- Block Size (Bytes) = 8
- Number of Blocks = 8

Obtemos:

- Hit Count: 14
- Accesses: 20
- Hit Rate: 0.7

Venus Editor Simulator Chocopy

Run Step Prev Reset Dump Trace Re-assemble from Editor

PC	Machine Code	Basic Code	Original Code
0x0	0x04C0006F	jal x0 76	j main
0x4	0xFF810113	addi x2 x2 -8	addi sp, sp, -8
0x8	0x00C12023	sw x12 0(x2)	sw a2, 0(sp)
0xc	0x00112223	sw x1 4(x2)	sw ra, 4(sp)
0x10	0x00200293	addi x5 x0 2	addi t0, x0, 2
0x14	0x005622B3	slt x5 x12 x5	slt t0, a2, t0
0x18	0x00028C63	beq x5 x0 24	beq t0, x0, anotherCall
0x1c	0x00412083	lw x1 4(x2)	lw ra, 4(sp)
0x20	0x00012603	lw x12 0(x2)	lw a2, 0(sp)
0x24	0x00810113	addi x2 x2 8	addi sp, sp, 8
0x28	0x00100513	addi x10 x0 1	addi a0, x0, 1
0x2c	0x00000067	jalr x0 x1 0	jr ra
0x30	0xFFFF00613	addi x12 x12 -1	addi a2, a2, -1

console output

Copy! Download! Clear!

Registers Memory **Cache** VDB

Cache Levels: 1

Block Size (Bytes): 8

Number of Blocks: 8

Associativity: 8

Cache Size (Bytes): 64

Enable? Enables current selected level of the cache.

Fully Associative

LRU L1

Hit Count: 14

Accesses: 20

Hit Rate: 0.7

0) HIT

1) HIT

2) HIT

3) HIT

4) HIT

5) HIT

6) EMPTY

7) EMPTY

NOTE: This is a write through, write allocate cache.

Seed: 6779040655319010227

Display Settings: Hex

Na análise comparativa entre o Mapeamento Direto e o Mapeamento Associativo, podemos observar o seguinte:

**1. Configuração com Block Size = 4 e Number of Blocks = 2:**

- Mapeamento Direto: Hit Rate de 0.1. A cache com mapeamento direto apresenta um desempenho inferior, pois apenas 10% das acessos foram bem-sucedidas. Isso ocorre porque a cache é dividida em blocos de tamanho 4 bytes e cada bloco é mapeado para uma posição específica na cache com base no seu endereço de memória.
- Mapeamento Associativo: Hit Rate de 0.1. Com o mapeamento associativo, o desempenho também é baixo, pois apenas 10% das acessos foram bem-sucedidas. No entanto, o mapeamento associativo permite que os blocos de dados sejam armazenados em qualquer posição da cache. Nesse caso, mesmo com a flexibilidade do mapeamento associativo, não houve melhoria significativa no desempenho em relação ao mapeamento direto.

**2. Configuração com Block Size = 8 e Number of Blocks = 2:**

- Mapeamento Direto: Hit Rate de 0.5. Com o aumento do tamanho do bloco para 8 bytes, houve uma melhoria significativa no desempenho da cache com mapeamento direto. O hit rate aumentou para 50%, o que significa que metade das acessos foram bem-sucedidas.
- Mapeamento Associativo: Hit Rate de 0.1. Mesmo com o aumento do tamanho do bloco, o mapeamento associativo não apresentou melhoria no desempenho em relação ao mapeamento direto. Ainda obteve um hit rate de 0.1, indicando que apenas 10% das acessos foram bem-sucedidas.

**3. Configuração com Block Size = 8 e Number of Blocks = 8:**

- Mapeamento Direto: Hit Rate de 0.7. Com o aumento do número de blocos para 8, o desempenho da cache com mapeamento direto melhorou significativamente. O hit rate aumentou para 70%, o que indica que 70% das acessos foram bem-sucedidas.
- Mapeamento Associativo: Hit Rate de 0.7. Na configuração com 8 blocos e mapeamento associativo, também obtivemos um hit rate de 0.7, o que significa que 70% das acessos foram bem-sucedidas. Isso indica que o mapeamento associativo foi capaz de aproveitar a flexibilidade do armazenamento de blocos em qualquer posição da cache e alcançar um desempenho equivalente ao mapeamento direto na mesma configuração.

Analisando os resultados, podemos concluir que o mapeamento associativo é capaz de alcançar um desempenho semelhante ao mapeamento direto quando o número de blocos na cache é suficientemente grande. No entanto, o mapeamento direto pode oferecer um desempenho melhor em configurações com menor número de blocos ou tamanhos de bloco menores.