Computer Organization

Lab14 Cache Types and Performance

Topics

- Cache: Types and Performance
  - Direct Mapped Cache
  - Fully Associative Cache
  - N-way Set Associative Cache

- To achive better cache performance
  - Suggestions on prgramming

Direct Map Cache performance(1)

```
.data
        array: .word 1,1,1
        tmp: .word 0 : 100
.text
        la $t0, array
        li $t1, 25
        loop:
                lw $t3, 0($t0)
                lw $t4, 4($t0)
                lw $t5, 8($t0)

                add $t2, $t3, $t4
                add $t2, $t2, $t5

                sw $t2, 12($t0)

                addi $t0, $t0, 16
                addi $t1, $t1, -1
                bgtz $t1, loop

        li $v0, 10
        syscall
```

- **512Byte =**

  32 Blocks * 4 words/every block * 4 Bytes/every word

  - There are totally 25 miss and 75 hit in 100 accessing, cache hit rate is **75%**.

- **512Byte =**

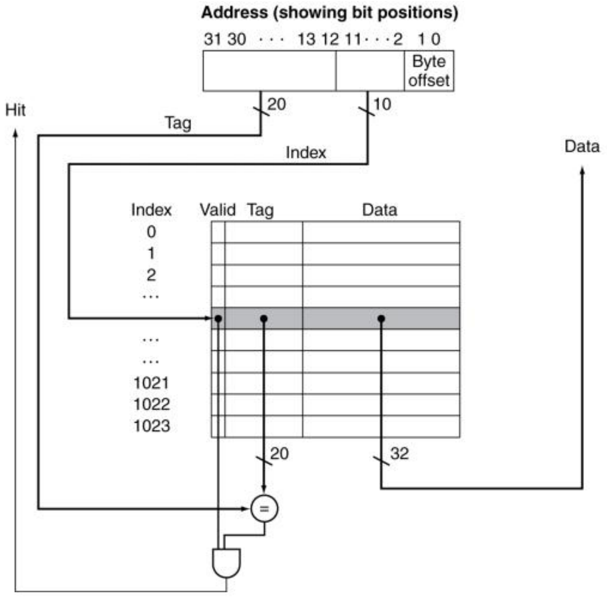
  16 Blocks * 8 words/every block * 4 Bytes/every word

  - There are totally 13 miss and 87 hit in 100 accessing, cache hit rate is **87%**.

Here **bigger** size of **cache block** lead to **higer cache hit rate**.

4 Direct Map Cache performance(2)

```
.data
        blk0: .word 0:32
        blk1: .word 0:32
.text
        add $t0,$0,$0
        add $s0,$0,$0
        addi $t1,$0,32
loop:
        lw $t2,blk0($t0)
        add $t2,$t2,$t0
        srl $t2,$t2,31
        sw $t2,blk1($t0)
        addi $t0,$t0,4
        addi $s0,$s0,1
        bne $s0,$t1,loop

        li $v0,10
        syscall
```



Q1. While running the demo on the MIPS CPU, How many time of memory access?

Q2. While there is a **Direct Map Cache(size：128Byte)** work with the CPU, what's the cache hit rate on the following settings?

Feature1)
ByteOffset： 2 bit-width
index： 5 bit-width

Feature2)
ByteOffset： 4 bit-width
index： 3 bit-width

5 Direct Map Cache performance(3)

```
.data
        blk0: .word 0:32
        blk1: .word 0:32
.text
        add $t0,$0,$0
        add $s0,$0,$0
        addi $t1,$0,32
loop:
        lw $t2,blk0($t0)
        add $t2,$t2,$t0
        srl $t2,$t2,31
        sw $t2,blk1($t0)
        addi $t0,$t0,4
        addi $s0,$s0,1
        bne $s0,$t1,loop

        li $v0,10
        syscall
```

Direct Map Cache

size： 128Byte

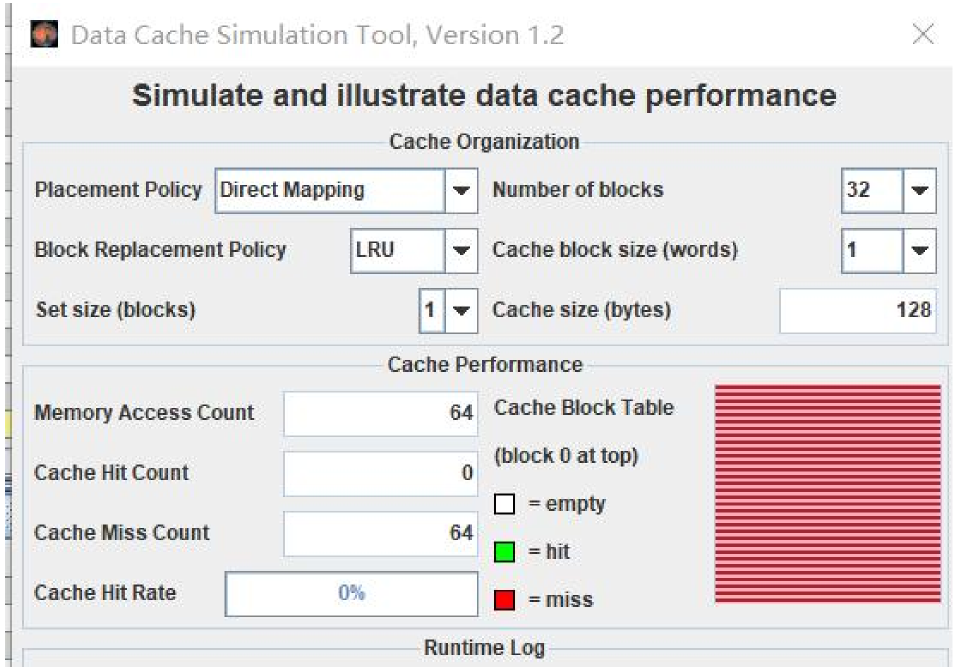
Feature1）

ByteOffset： 2 bit-width

Index： 5 bit-width

cache hit rate is 0!!

Would wider the size of cache block bring better cache hit rate?



Data Cache Simulation Tool, Version 1.2

Simulate and illustrate data cache performance

Cache Organization

Placement Policy: Direct Mapping

Number of blocks: 32

Block Replacement Policy: LRU

Cache block size (words): 1

Set size (blocks): 1

Cache size (bytes): 128

Cache Performance

Memory Access Count: 64

Cache Hit Count: 0

Cache Miss Count: 64

Cache Hit Rate: 0%

Cache Block Table

(block 0 at top)

□ = empty

🟩 = hit

🟥 = miss

Runtime Log

6 Direct Map Cache performance(3)

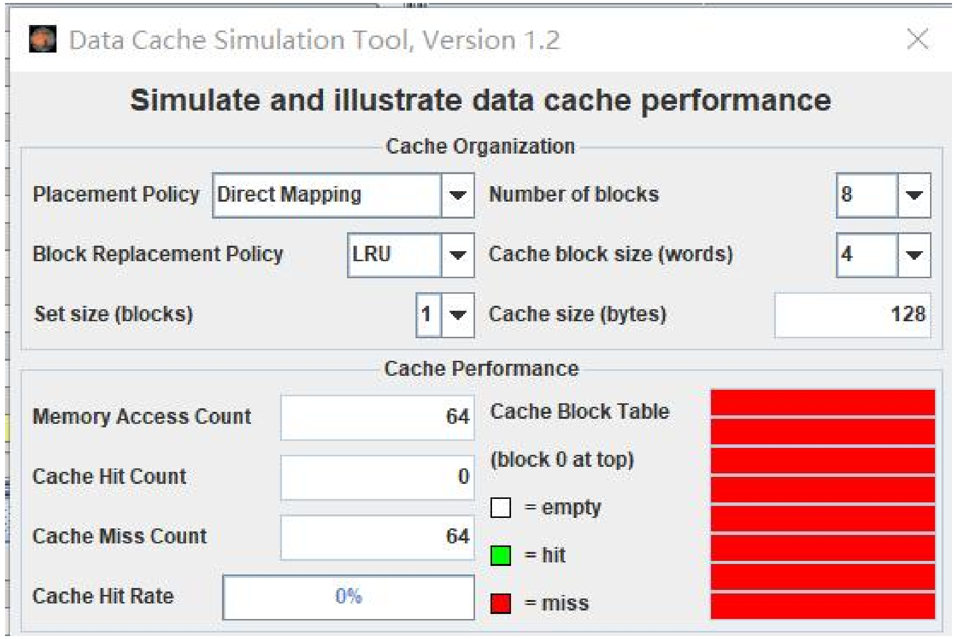
```
.data
	blk0: .word 0:32
	blk1: .word 0:32
.text
	add $t0,$0,$0
	add $s0,$0,$0
	addi $t1,$0,32
loop:
	lw $t2,blk0($t0)
	add $t2,$t2,$t0
	srl $t2,$t2,31
	sw $t2,blk1($t0)
	addi $t0,$t0,4
	addi $s0,$s0,1
	bne $s0,$t1,loop

	li $v0,10
	syscall
```

Direct Map Cache
size：128Byte

Feature2)
ByteOffset： 4 bit-width
Index： 3 bit-width

cache hit rate is 0!!

Would wider the size of cache block bring better cache hit rate?



Data Cache Simulation Tool, Version 1.2

Simulate and illustrate data cache performance

Cache Organization

Placement Policy: Direct Mapping
Number of blocks: 8
Block Replacement Policy: LRU
Cache block size (words): 4
Set size (blocks): 1
Cache size (bytes): 128

Cache Performance

Memory Access Count: 64
Cache Hit Count: 0
Cache Miss Count: 64
Cache Hit Rate: 0%

Cache Block Table
(block 0 at top)
□ = empty
🟩 = hit
🟥 = miss

Fully associative (1)

```
.data
	blk0: .word 0:32
	blk1: .word 0:32
.text
	add $t0,$0,$0
	add $s0,$0,$0
	addi $t1,$0,32
loop:
	lw $t2,blk0($t0)
	add $t2,$t2,$t0
	srl $t2,$t2,31
	sw $t2,blk1($t0)
	addi $t0,$t0,4
	addi $s0,$s0,1
	bne $s0,$t1,loop

	li $v0,10
	syscall
```

- Fully associative Cache
  - **Allow a given block to go in ANY cache entry**
  - Requires all entries to be searched at once
  - Comparator per entry

Q1. While there is a **Fully associative Cache(size：128Byte)** work with the CPU, what’s the cache hit rate on the following settings?

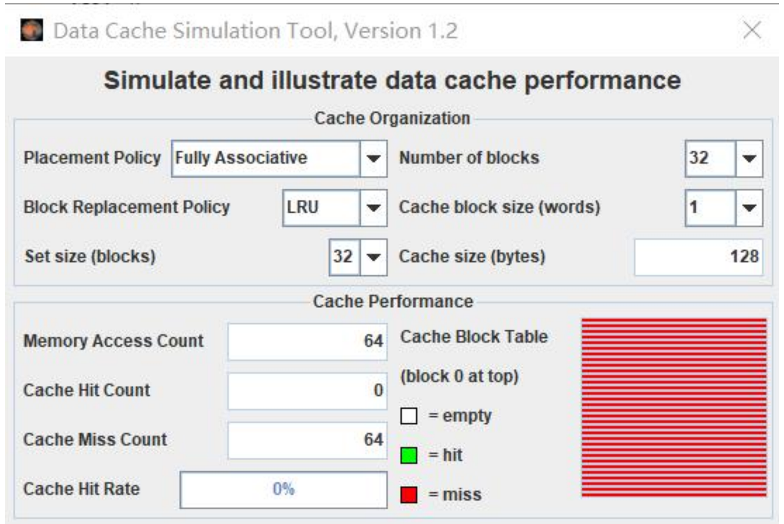
Feature1			Feature2	
ByteOffset	2 bit-width		ByteOffset	4 bit-width
Index	5 bit-width		Index	3 bit-width

8 Fully associative (2)

```
.data
        blk0: .word 0:32
        blk1: .word 0:32
.text
        add $t0,$0,$0
        add $s0,$0,$0
        addi $t1,$0,32
loop:
        lw $t2,blk0($t0)
        add $t2,$t2,$t0
        srl $t2,$t2,31
        sw $t2,blk1($t0)
        addi $t0,$t0,4
        addi $s0,$s0,1
        bne $s0,$t1,loop

        li $v0,10
        syscall
```

Would Fully associative cache bings higher cache hit rate?

Fully associative Cache
size： 128Byte
Feature1)
ByteOffset： 2 bit-width
Index： 5 bit-width



Data Cache Simulation Tool, Version 1.2

Simulate and illustrate data cache performance

Cache Organization

Placement Policy: Fully Associative
Number of blocks: 32
Block Replacement Policy: LRU
Cache block size (words): 1
Set size (blocks): 32
Cache size (bytes): 128

Cache Performance

Memory Access Count: 64
Cache Hit Count: 0
Cache Miss Count: 64
Cache Hit Rate: 0%

Cache Block Table
(block 0 at top)
□ = empty
🟩 = hit
🟥 = miss

cache hit rate is 0!!

Would wider the size of cache block bring better cache hit rate in the cache?

Fully associative (3)

```
.data
        blk0: .word 0:32
        blk1: .word 0:32
.text
        add $t0,$0,$0
        add $s0,$0,$0
        addi $t1,$0,32
loop:
        lw $t2,blk0($t0)
        add $t2,$t2,$t0
        srl $t2,$t2,31
        sw $t2,blk1($t0)
        addi $t0,$t0,4
        addi $s0,$s0,1
        bne $s0,$t1,loop

        li $v0,10
        syscall
```

Would Fully associative cache bings higher cache hit rate?

Direct Map Cache

size：128Byte

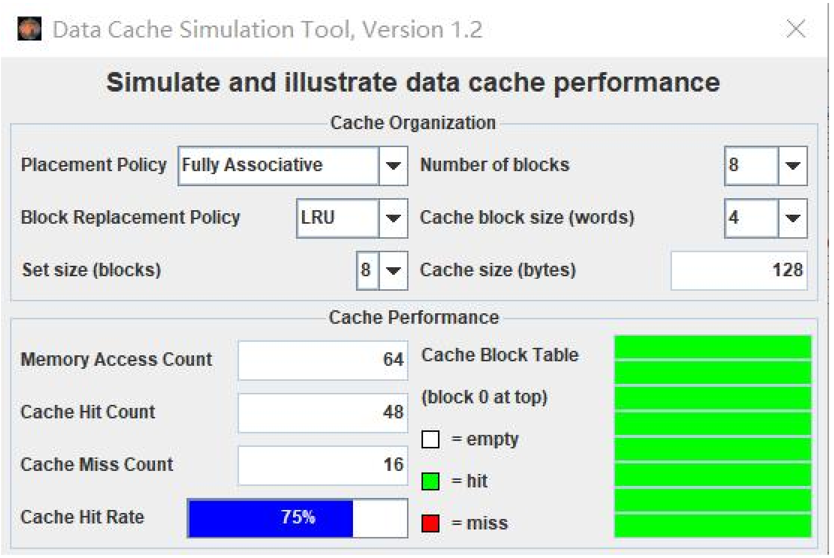
Feature2）

ByteOffset：　4 bit-width

Index：　　3 bit-width

cache hit rate is 75%!!

Would wider the size of cache block bring better cache hit rate in the cache?



Data Cache Simulation Tool, Version 1.2

Simulate and illustrate data cache performance

Cache Organization

Placement Policy: Fully Associative

Number of blocks: 8

Block Replacement Policy: LRU

Cache block size (words): 4

Set size (blocks): 8

Cache size (bytes): 128

Cache Performance

Memory Access Count: 64

Cache Hit Count: 48

Cache Miss Count: 16

Cache Hit Rate: 75%

Cache Block Table (block 0 at top)

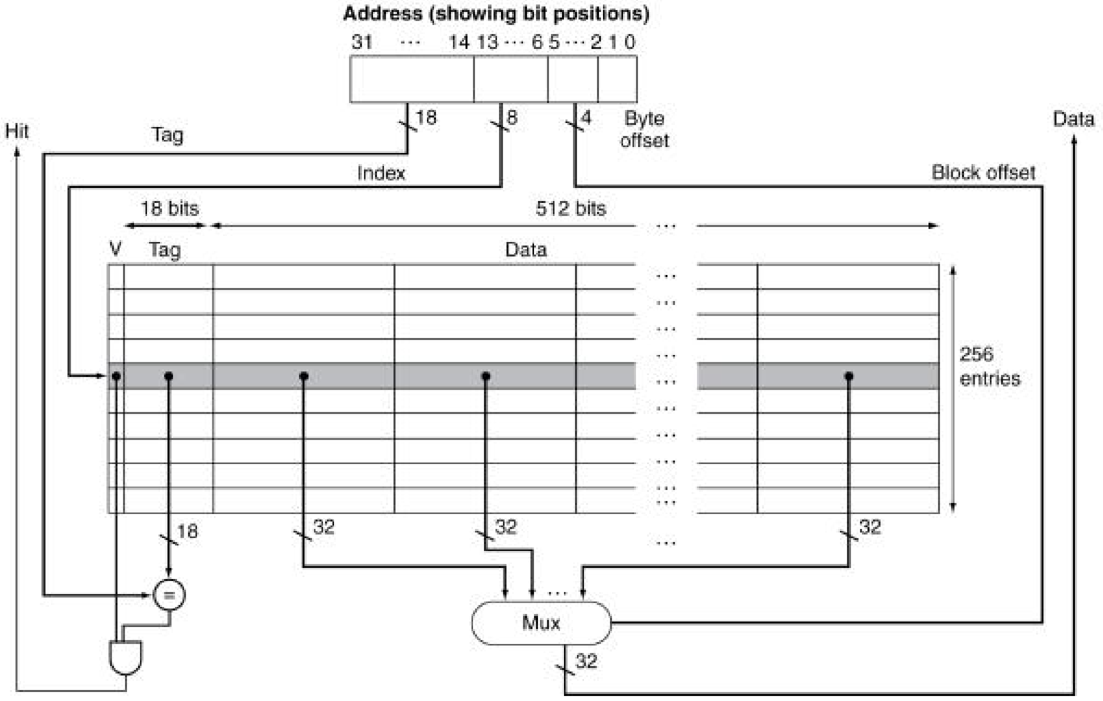
□ = empty

🟩 = hit

🟥 = miss

10 N-way Set Associative Cache(1)

- N-way set associative Cache
  - **Each set contains n entries**
  - Block number determines which set
    - (Block number) modulo (#sets in cache)
  - **Search all entries in a given set at once**
  - **n comparators**



Fully associative <- N-way Set associative -> Direct Mapping

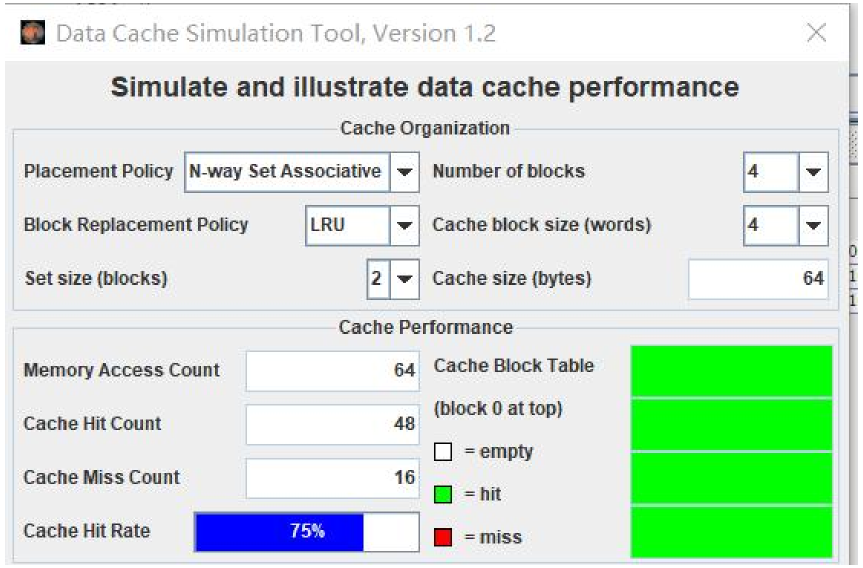
11 N-way Set Associative Cache(2)

```
.data
        blk0: .word 0:32
        blk1: .word 0:32
.text
        add $t0,$0,$0
        add $s0,$0,$0
        addi $t1,$0,32
loop:
        lw $t2,blk0($t0)
        add $t2,$t2,$t0
        srl $t2,$t2,31
        sw $t2,blk1($t0)
        addi $t0,$t0,4
        addi $s0,$s0,1
        bne $s0,$t1,loop

        li $v0,10
        syscall
```

N-way set associative Cache
Total size： 128Byte
Feature1）
2-way set associative
ByteOffset： 4 bit-width
Index： 3 bit-width

cache hit rate is 75%!!

Would wider the size of cache block bring better cache hit rate in the cache?



Data Cache Simulation Tool, Version 1.2

Simulate and illustrate data cache performance

Cache Organization

Placement Policy: N-way Set Associative
Number of blocks: 4
Block Replacement Policy: LRU
Cache block size (words): 4
Set size (blocks): 2
Cache size (bytes): 64

Cache Performance

Memory Access Count: 64
Cache Hit Count: 48
Cache Miss Count: 16
Cache Hit Rate: 75%

Cache Block Table
(block 0 at top)
□ = empty
🟩 = hit
🟥 = miss

12 N-way Set Associative Cache(3)

```
.data
        blk0: .word 0:32
        blk1: .word 0:32
.text
        add $t0,$0,$0
        add $s0,$0,$0
        addi $t1,$0,32
loop:
        lw $t2,blk0($t0)
        add $t2,$t2,$t0
        srl $t2,$t2,31
        sw $t2,blk1($t0)
        addi $t0,$t0,4
        addi $s0,$s0,1
        bne $s0,$t1,loop

        li $v0,10
        syscall
```

N-way set associative Cache

Total size： 128Byte

Feature1)

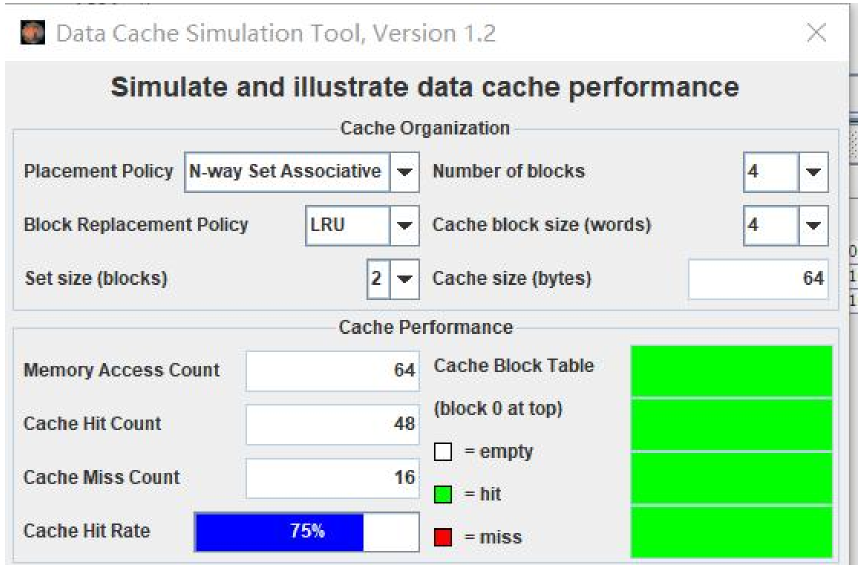
2-way set associative

ByteOffset： 4 bit-width

Index： 3 bit-width

cache hit rate is 75%!!

Would wider the size of cache block bring better cache hit rate in the cache?



Data Cache Simulation Tool, Version 1.2

Simulate and illustrate data cache performance

Cache Organization

Placement Policy	N-way Set Associative	Number of blocks	4
Block Replacement Policy	LRU	Cache block size (words)	4
Set size (blocks)	2	Cache size (bytes)	64

Cache Performance

Memory Access Count	64
Cache Hit Count	48
Cache Miss Count	16
Cache Hit Rate	75%

Cache Block Table

(block 0 at top)

□ = empty

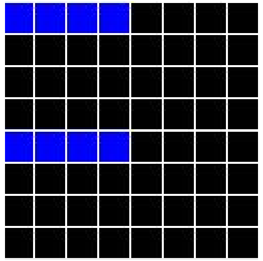
🟩 = hit

🟥 = miss

13 Achive better cache performance by programming(1)

Demo1

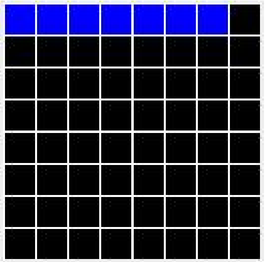
```
int a[size];
int b[size];
```



```
.data
        blk0: .word 0:32
        blk1: .word 0:32
.text
        add $t0,$0,$0
        add $s0,$0,$0
        addi $t1,$0,32
loop:
        lw $t2,blk0($t0)
        add $t2,$t2,$t0
        srl $t2,$t2,31
        sw $t2,blk1($t0)
        addi $t0,$t0,4
        addi $s0,$s0,1
        bne $s0,$t1,loop

        li $v0,10
        syscall
```

Demo2

```
struct merge{
int a;
int b;
};
struct merge marr[size];
```



Which one has better cache pergormance? Demo1 or Demo2? Why?

```
.data
        mblk: .word 0:64
.text
        add $t0,$0,$0
        add $s0,$0,$0
        addi $t1,$0,32
loop:
        lw $t2,mblk($t0)
        add $t2,$t2,$t0
        srl $t2,$t2,31
        addi $t0,$t0,4
        sw $t2,mblk($t0)
        addi $t0,$t0,4
        addi $s0,$s0,1
        bne $s0,$t1,loop

        li $v0,10
        syscall
```

Achive better cache performance by programming(2)

Demo1

```
for(i=0;i<size;i++)
        B[i] = A[i];
for(i=0;i<size;i++)
        C[i] = A[i];
```

```
.data  #piece 1
        blk0: .word 0:32
        blk1: .word 0:32
        blk2: .word 0:32
.text
        add $t0,$0,$0
        add $s0,$0,$0
        addi $t1,$0,32
loop:
        lw $t2,blk0($t0)
        add $t2,$t2,$t0
        srl $t2,$t2,31

        sw $t2,blk1($t0)

        addi $t0,$t0,4
        addi $s0,$s0,1
        bne $s0,$t1,loop
```

```
loop2:  #piece2
        lw $t2,blk0($t0)
        add $t2,$t2,$t0
        srl $t2,$t2,31

        sw $t2,blk2($t0)

        addi $t0,$t0,4
        addi $s0,$s0,1
        bne $s0,$t1,loop

        li $v0,10
        syscall
```

Demo2

```
for(i=0;i<size;i++){
        B[i] = A[i];
        C[i] = A[i];
}
```

Which one has better cache pergormance? Demo1 or Demo2? Why?

```
.data
        blk0: .word 0:32
        blk1: .word 0:32
        blk2: .word 0:32
.text
        add $t0,$0,$0
        add $s0,$0,$0
        addi $t1,$0,32
loop:
        lw $t2,blk0($t0)
        add $t2,$t2,$t0
        srl $t2,$t2,31

        sw $t2,blk1($t0)
        sw $t2,blk2($t0)

        addi $t0,$t0,4
        addi $s0,$s0,1
        bne $s0,$t1,loop

        li $v0,10
        syscall
```

15 Achive better cache performance by programming(3)

```
.data   #Demo1P1/2
 # 32*2 word (rows: 32, lines: 2)
matrix: .space 256

.macro getindex(%ans,%i,%j)
        sll %ans,%i,complete here
        add %ans,%ans,%j
        sll %ans,%ans,complete here
.end_macro

.text
addi  $t0,$0,0  #i
addi  $s0,$0,2

addi  $t1,$0,0  #j
addi $s1,$0,32
```

```
loopi:    #Demo1P2/2
beq $t0,$s0,loopiend

addi  $t1,$0,0

loopj:
beq $t1,$s1,loopjend
getindex($a0,$t0,$t1)
lw $v0,matrix($a0)
addi $t1,$t1,1
j loopj

loopjend:
addi $t0,$t0,1
j loopi

loopiend:
li $v0,10
syscall
```

Demo1

```
int matrix[2][32];


for( i=0;i<2;i++ ){
        for( int j=0;j<32;j++ )
                matrix[i][j] ...
}
```

Achive better cache performance by programming(4)

```
.data   #Demo2P1/2
 # 32*2 word (rows: 32, lines: 2)
matrix: .space 256

.macro getindex(%ans,%i,%j)
        sll %ans,%i,complete here
        add %ans,%ans,%j
        sll %ans,%ans,complete here
.end_macro

.text
addi  $t0,$0,0  #i
addi  $s0,$0,2

addi  $t1,$0,0  #j
addi $s1,$0,32
```

```
loopj:   #Demo1P2/2
beq $t1,$s1,loopjend

addi  $t0,$0,0
loopi:
beq $t0,$s0,loopiend
getindex($a0,$t0,$t1)
lw $v0, matrix($a0)
addi $t0,$t0,1
j loopi

loopiend:
addi $t1,$t1,1
j loopj

loopjend:
li $v0,10
syscall
```

Demo2

```
int matrix[2][32];


for( j=0;j<32;j++ ){
        for( int i=0;i<2;i++ )
                matrix[i][j] ...
}
```

Which one has better cache pergormance? Demo1 or Demo2?
Why?