Kingdom of Saudi Arabia
Ministry of Education
University of Jeddah
College of Computer Science & Engineering
Department of Computer Science & AI

المملكة العربية السعودية
وزارة التعليم
جامعة جدة
كلية علوم و هندسة الحاسب
قسم علوم الحاسب الآلي و الذكاء الاصطناعي

CCCS217 Computer Organization and Architecture

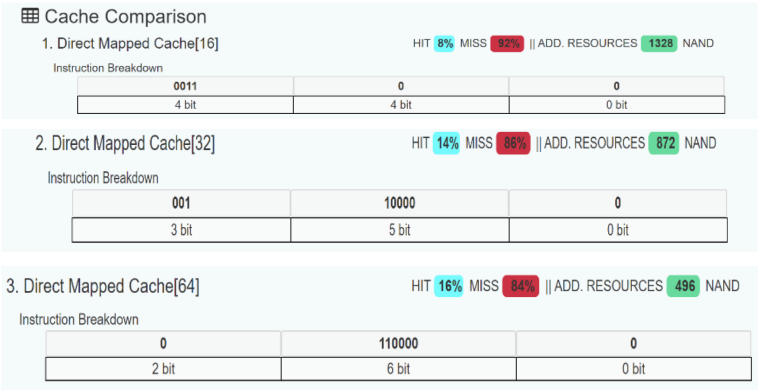
Course Project

Students' names:

Project Description:

Part1: is related to mapping of memory to cache

1. For All caches (Direct Mapping)



a. Set the size of Cache so that Direct Mapping has the maximum hit ratio and minimum miss ratio?

64 size of cache has maximum hit ratio: 16%, and minimum miss ratio: 84%.

16 size of cache has minimum hit ratio: 8%, and maximum miss ratio: 92%.

b. How can you increase the hit ratio in case of direct mapping by changing some of size of Caches?

We can increase the hit ratio in direct mapping by increasing the cache size.

c. In presentation, show the pictures from the simulator containing the values used and results obtained. Also, give reasons why the hit ratio increases when you change the size of caches?

When the size of the cache increases, then number of blocks will increase. Since there are many blocks, many pages from main memory will gets accommodate in the cache memory. If take more blocks are present in cache memory, then probability of finding a word in the cache will get increase. Hence hit ratio increase.

If there are " n " blocks in cache initially, it will be "n+m" blocks after increase in the size.

d. Explain the reason for the number of bits being used for Tag and for the number of bits required for RAM address?

1 -To find the number of bits required for main memory address:

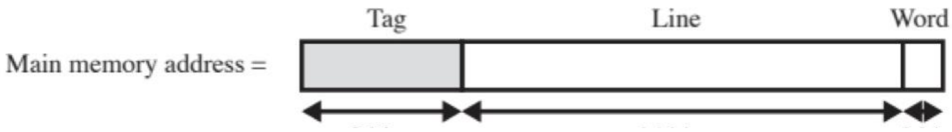
Simply by adding tag bits and bits of lines with the block offset

It's also knowing as "Physical Address Space"

The formula:

PAS = tag + line + block offset

As in the picture the PAS was 258 B --> number of bits = $\lceil \log_2 256 \rceil$ = 8 bits



Since the block offset is 0, line size = block size

the number of lines = cash size / line size

the number of lines = $2^4 / 2^0 = 2^4$

number of bits = $\lceil \log_2 2^4 \rceil$ = 4 bits

2-To find tag bits :

Tag = PAS – bits of lines

Tag = 8 – 4 --> Tag = 4 bits

2. For All caches (Fully Associative Mapping)

a. Set the size of Cache so that Fully Associative Mapping has the maximum hit ratio and minimum miss ratio?

64 size of cache has maximum hit ratio: 13%, and minimum miss ratio: 88%.

16 size of cache has minimum hit ratio: 8%, and maximum miss ratio: 92%.

b. How can you increase the hit ratio in case of Fully Associative mapping by changing some of size of Caches?

When we increase the size to bigger size, the hit ratio will be higher.

c. In presentation, show the pictures from the simulator containing the values used and results obtained. Also, give reasons why the hit ratio increases when you change the size of caches?

A block of main memory can present any where in the cache. Due to increase in space more blocks will gets accommodated.

d. Explain the reason for the number of bits being used for Tag and for the number of bits required for RAM address?

1-To find the number of bits required for main memory address:

Simply by adding tag bits with the block offset

It’s also knowing as “Physical Address Space”

Because there’s only tag and block offset

The formula:

PAS = tag + block offset

As in the picture the PAS was 258 B --> number of bits = $\lceil \log_2 256 \rceil$ = 8 bits

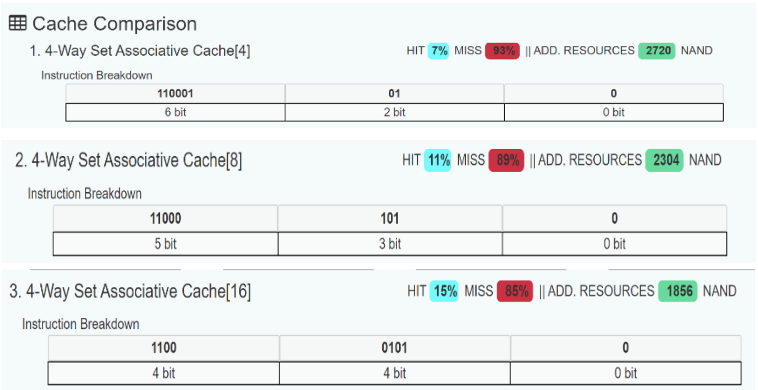
2-To find tag bits :

Since the block offset is 0

Tag = PAS

Tag = 8 bits

3. For All caches (4-way Associative Mapping)



a. Set the size of Cache so that 4-way Associative Mapping has the maximum hit ratio and minimum miss ratio?

16 size of cache has maximum hit ratio: 13%, and minimum miss ratio: 88%.

4 size of cache has minimum hit ratio: 8%, and maximum miss ratio: 92%.

b. How can you increase the hit ratio in case of 4-way Associative mapping by changing some of size of Caches?

When the size increasing, the hits will be increasing and miss hits will decreasing.

c. In presentation, show the pictures from the simulator containing the values used and results obtained. Also, give reasons why the hit ratio increases when you change the size of caches?

Due to increase in size, number of sets increases, but set size remains same. Since there are many sets chance of finding word is high.

Decreasing in cache size reduces hit rate, as there are only blocks.

d. Explain the reason for the number of bits being used for Tag and for the number of bits required for RAM address?

1-To find the number of bits required for main memory address:

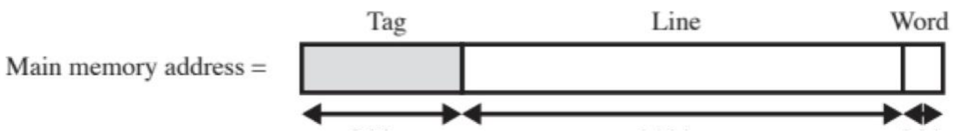
Simply by adding tag bits and bits of sets with the block offset

It’s also knowing as “Physical Address Space”

The formula:

PAS = tag + set + block offset

As in the picture the PAS was 258 B --> number of bits = $\lceil \log_2 256 \rceil$ = 8 bits



Since the block offset is 0

the number of sets = number of ways or sets (v)

the number of sets = 4 B

number of bits = $\lceil \log_2 4 \rceil$ = 2 bits

2-To find tag bits :

Tag = PAS – bits of sets

Tag = 8 – 2 --> Tag = 6 bits

Part2: Write an Assembly Language Program

This assembly code will convert an octal number to a decimal number and will make sure that the input is between 0-7 only. By using two loops, one if condition and two functions.

Edit | Execute

Project (1).asm

```
.data
message1: .asciiz "Rama Alyoubi Raghad Almutari Ryouf Alghamdi \n\n"
EnterMessage: .asciiz "Enter three digit of octal number\t"
above7message: .asciiz "One of the digits was above 7\nPlease enter a valid octal number\t"
invalidMessage: .asciiz "invalid octal number please try again\t"
result: .asciiz "The result:\t"


.text
main:

la $a0, message1                        # print message
li $v0, 4
syscall
                                        # print message
la $a0, EnterMessage
li $v0, 4
syscall

jal input                               # calling input function to take integer value from user


while1:
        li $t1,0                # Load immediately $t1=0
        li $t2,777              # Load immediately $t2=777
        sgt $t3,$t0,$t1         # is input grater than 0? , %t3 = boolean excprtion (true or false) (1,0)
        slt $t4,$t0,$t2         # is input less than 777? , %t4 = boolean excprtion (true or false) (1,0)
        and $t5,$t4,$t3         # condition 1 AND condition 2 (to make it OR)
        beq $t5,1,exit                  # branch to exit if true

        la $a0, invalidMessage          # print message
        li $v0, 4
        syscall
        jal input                       # calling input function to take integer value from user
        move $t0, $v0

        j while1                # jump to the loop function untel the condition in branch statement is tr
        exit:                           # end of loop

        move $t7, $t0
while2:
        li $t6,10                       # Load immediately $t6 = 10
        li $t3,0                # Load immediately $t3 = 0
        rem $t3,$t7,$t6                 # $t7 = input % $t6 (save the remainder in $t7)
        li $t8,8                        # Load immediately $t8 = 8
        slt $t2,$t3,$t8                 # is $t2 less than 8? , $t2 = boolean excprtion (true or false) (
        beq $t2,1,else                  # branch to else if true
        if:
        la $a0, above7message           # print message
        li $v0, 4
        syscall
       jal input                        # calling input function to take integer value from user
       move $t7, $v0
       j while2
        else:
        div $t7,$t7,$t6
        beq $t7,0,exit2                 # branch to exit if true

        j while2                # jump to the loop function untel the condition in branch statement is tr

exit2:                                  # end of loop


li $a1,0                                # Load immediately $a1 = 0 (the power)
li $a0,8                                # Load immediately $a0 = 8 (the base)

jal power                               # calling pow function
move $t2, $v0                           # return the value from the power function and move it to $t2


rem $t3,$t0,10                          # $t3 = input % $t6 (save the remainder in $t3)
mul $t4,$t3,$t2                 # $t4 = $t3 * $t2 (save the multiplication in $t4)
add $t1,$t1,$t4                 # $t1 = $t1 + $t4 (save the addition in $t1)
div $t0,$t0,10                          # $t0 = $t0 / 10 (save the division in $t0)
```

```
li $a0,8                                  # Load immediately $a0 = 8 (the base)
li $a1,1                                  # Load immediately $a1 = 1 (the power)
jal power                                 # calling pow function
move $t5, $v0                             # return the value from the power function and move it to $t5

rem $t3,$t0,10                            # $t3 = input % $t6 (save the remainder in $t3)
mul $t4,$t3,$t5                   # $t4 = $t3 * $t2 (save the multiplication in $t4)
add $t1,$t1,$t4                   # $t1 = $t1 + $t4 (save the addition in $t1)
div $t0,$t0,10                            # $t0 = $t0 / 10 (save the division in $t0)

li $a0,8                                  # Load immediately $a0 = 8 (the base)
li $a1,2                                  # Load immediately $a1 = 2 (the power)
jal power                                 # calling pow function
move $t7, $v0                             # return the value from the power function and move it to $t7

rem $t3,$t0,10                            # $t3 = input % $t6 (save the remainder in $t3)
mul $t4,$t3,$t7                   # $t4 = $t3 * $t2 (save the multiplication in $t4)
add $t1,$t1,$t4                   # $t1 = $t1 + $t4 (save the addition in $t1)
div $t0,$t0,10                            # $t0 = $t0 / 10 (save the division in $t0)



la $a0, result                            # print message
li $v0, 4
syscall

move $a0, $t1

li $v0, 1                                 # print the value
syscall


li $v0,10                                 # end of main
syscall

input:
                                          # take input from the user
        li $v0, 5
        syscall
                                          # move to a temp regester
        move $t0, $v0
         jr $ra                   # return value

power:
        bne $a1, $zero, recursion       # if the power is greater then 1,then do some recursion
        li  $v0,1                       # otherwise, return 1.
        jr  $ra                           # Return to the calling function
recursion:
        addi $sp, $sp, -4               # Allocate space for one integer on the stack
        sw   $ra,0($sp)                 # Store the return address on the stack
        addi $a1,$a1,-1                 # Decrement the power by 1.
        jal  power                      # Call the power function with the new parameters
        mul  $v0, $a0, $v0              # Multiply the result by the base and save it as the new result
        lw   $ra, 0($sp)                # Restore the return address from the stack
        addi $sp,$sp,4                  # Deallocate the memory on the stack
        jr   $ra                        # Return to the calling function
```

$zero	0	0x00000000
$at	1	0x00000000
$v0	2	0x00000000
$v1	3	0x00000000
$a0	4	0x00000000
$a1	5	0x00000000
$a2	6	0x00000000
$a3	7	0x00000000
$t0	8	0x00000000
$t1	9	0x00000000
$t2	10	0x00000000
$t3	11	0x00000000
$t4	12	0x00000000
$t5	13	0x00000000
$t6	14	0x00000000
$t7	15	0x00000000
$s0	16	0x00000000
$s1	17	0x00000000
$s2	18	0x00000000
$s3	19	0x00000000
$s4	20	0x00000000
$s5	21	0x00000000
$s6	22	0x00000000
$s7	23	0x00000000
$t8	24	0x00000000
$t9	25	0x00000000
$k0	26	0x00000000
$k1	27	0x00000000
$gp	28	0x10008000
$sp	29	0x7fffeffc
$fp	30	0x00000000
$ra	31	0x00000000
pc		0x00400000
hi		0x00000000
lo		0x00000000

Mars Messages **Run I/O**

```
Rama Alyoubi Raghad Almutari Ryouf Alghamdi

Enter three digit of octal number       -1
invalid octal number please try again   290
One of the digits was above 7
Please enter a valid octal number       123
The result:     83
-- program is finished running --
```