**Week 1: Computer performance and**

**MIPS Instruction Set Architecture**

**Name: Lâm Nhật Tân**

**MSSV: 2010597**

**2. Quiz**

**2.1.**

1. What is CPU execution time? Distinguish system performance and CPU performance.
2. What is CPU time equation? (Write the equation in all forms from general to detailed)
3. What hardware/software components affect a program's performance (affect the factors in the CPU performance equation)?

**Answer:**

1. Is the total time a CPU spends computing on a given task and does not include time spent waiting for I/O or running other programs.
2. CPU execution time for a program = CPU clock cycles x Clock cycle time

=

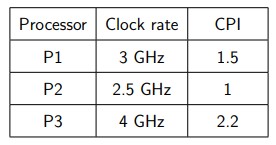
= Instruction count x CPI x Clock cycle time

=

1. There are many things that affect a program's performance.

Ex: **compiler**, **programming language**, and **algorithm** affect instruction count, **CPI**, **Instruction set architecture**, **clock rate**…

**2.2.** Consider three different processors P1, P2 and P3 executing the same instruction net.



1. Which processor has the highest performance expressed in instructions per second?
2. If the processors each execute a program in 10 seconds, find the number of cycles and the number of instructions.
3. We are trying to reduce the execution time by 30%, but this leads to an increase of 20% in the CPI. What clock rate should we have to get this time reduction?

**Answer:**

1. CPU time =

Because P1, P2, P3 execute the same instruction net, so we compare their performance by CPI/Clock rate. So the P2 processor has the highest performance.

1. CPU time = 10s

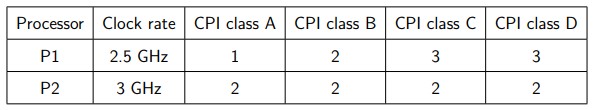
|  |  |  |
| --- | --- | --- |
| Processor | Number of cycles | Number of instructions |
| P1 | 3 \* 1010 | 2 \* 1010 |
| P2 | 2.5 \* 1010 | 2.5 \* 1010 |
| P3 | 4 \* 1010 | 1.81 \* 1010 |

1. CPU time =

⇒ x = 171.42%

So we should increase of 71,42% of clock rate to get that time reduction.

**2.3.**Consider two different implementations of the same instruction set architecture. The instructions can be divided into four classes according to their CPI (classes A, B, C, and D).



1. Given a program with a dynamic instruction count of 106 instructions divided into classes as follows: 10% class A, 20% class B, 50% class C, and 20% class D, which processor is faster?
2. Find the average CPI for each implementation.
3. Find the total clock cycles required in both cases.

**Answer:**

1. CPI of P1 = 0.1 \* 1 + 0.2 \* 2 + 0.5 \* 3 + 0.2 \* 3 = 2.6

CPI of P2 = 0.1 \* 2 + 0.2 \* 2 + 0.5 \* 2 + 0.2 \* 2 = 2

The processor P2 is the faster.

1. CPU time of P1 = (106 \* 2.6) / (2.5 \* 109) = 1.04 \* 10-3 (s)

CPU time of P2 = (106 \* 2) / (3 \* 109) = 0.66 \* 10-3 (s)

1. CPU clock cycles for a program = Instruction count \* CPI

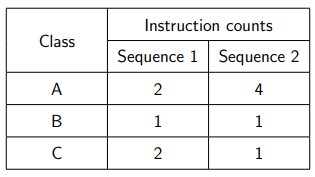
|  |  |  |
| --- | --- | --- |
| Processor | Instruction count | Clock cycles |
| P1 | 1 \* 106 | 2.6 \* 106 |
| P2 | 1 \* 106 | 2 \* 106 |

**2.4.** A given application written in Java runs 15 seconds on a desktop processor. A new Java compiler is released that requires only 0.6 as many instructions as the old compiler. Unfortunately, it increases the CPI by 1.1. How fast can we expect the application to run using this new compiler?

**Answer:** CPU time (new) = CPU time (given) \* 1.1 \* 0.6 = 15 \* 1.1 \* 0.6 = 9.9 (seconds)

# 2.5 Question 5

Two compiler complied a program into 2 different code sequences that require the instruction counts as shown in below table. Know that the CPI of instructions class A is 1, class B is 2, class C is 3.



1. Which code sequence executes the most instructions?
2. Which will be faster?
3. What is the CPI for each sequence?

**Answer:**

1. Sequence 2 executes 4 + 1 + 1 = 6 instruction

Sequence 1 executes:2 + 1 + 2 = 5 instruction

=> Sequence 2 executes the most instructions.

1. Sequence 1 will be faster.

c. + Sequence 1: CPI = 2

+ Sequence 2: CPI = 1.5

# 2.6 Question 6

1. What are 32-bit MIPS instruction formats? Explain fields of these formats.
2. Encode this MIPS instruction to machine binary, then rewrite it in hex representation:

sw $t1, 32($t2)

1. Which MIPS code represents the following machine binary code? 00000010001100100100000000100000

**Answer:**

1. Three instruction categories: I-format, J-format, and R-format.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name | Layout | | | | | |
| Size | 6 bits | 5 bits | 5 bits | 5 bits | 5 bits | 6 bits |
| R-format | op | rs | rt | rd | shamt | funct |
| I-format | op | rs | rt | address/ immediate | | |
| J-format | op | target address | | | | |

-op (opcode): basic operation of the instruction.

-rs: the first register source operand.

-rt: the second register source operand.

-rd: the register destination operand, which gets the result of the operation.

-shamt: shift amount to be used in shift instructions, zero otherwise.

-funct: called the function code.

1. Encode : **sw $t1, 32($t2)**

Binary: **101011 01010 01001 0000000000100000**

Hex: **0xAD49 0020**

1. MIPS code represents the following machine binary code:

**add $t0, $s1, $s2**

**3. Exercise**

**3.1.** Write a simple MIPS program that can execute these steps:

1. Print a sentence to terminal to request an integer number from user;

2. Collect the number and increase it by 1;

3. Print the result to terminal.

|  |
| --- |
| .data  mess : .asciiz "Enter value: "  .text  main:  li $v0, 4  la $a0, mess  syscall    li $v0, 5  syscall    move $t0, $v0  add $t0, $t0, 1    li $v0, 1  move $a0, $t0  syscall    li $v0, 10  syscall |
|  |

**3.2.**

Write a small program that allows users to input values for variables a, b, c, and d. The program then calculates the following expressions and prints the results to terminal. g = (a + b) × 3 − (c + d) × 2

|  |  |
| --- | --- |
| .data  messa: .asciiz "Give value a = "  messb: .asciiz "Give value b = "  messc: .asciiz "Give value c = "  messd: .asciiz "Give value d = "  result: .asciiz "g = (a + b) \* 3 - (c + d) \* 2 = "  .text  main:  #a  li $v0, 4  la $a0, messa  syscall  li $v0, 5  syscall  move $t0, $v0  #b  li $v0, 4  la $a0, messb  syscall  li $v0, 5  syscall  move $t1, $v0  #c  li $v0, 4  la $a0, messc  syscall  li $v0, 5  syscall  move $t2, $v0 | #d  li $v0, 4  la $a0, messd  syscall  li $v0, 5  syscall  move $t3, $v0  #res = $t4  add $t4, $t0, $t1 # (a+b) = $t4  mul $t4, $t4, 3 # (a+b)\*3    add $t5, $t2, $t3 # (c+d) = $t5  mul $t5, $t5, 2    sub $t4, $t4, $t5 #res    #print  li $v0, 4  la $a0, result  syscall  li $v0, 1  move $a0, $t4  syscall    li $v0, 10  syscall |
|  | |

**3.3** Write a MIPS program with the following requirements:

1. Declare an integer array with 10 synthetic data elements.

2. Calculate the sum of all array elements.

3. Print the result to the terminal.

|  |
| --- |
| .data  mess: .asciiz "result sum 1->10 = "  array: .word 1,2,3,4,5,6,7,8,9,10  .text  main:  li $v0, 0 #i = 0  li $t4, 10  la $s1, array  li $t0, 0  FOR\_LOOP:  bge $s0, $t4, END\_FOR  lw $t1, ($s1)  add $t0, $t0, $t1  addi $s1, $s1, 4  addi $s0, $s0, 1  j FOR\_LOOP  END\_FOR:  addi $s0, $s0, -1  addi $s1, $s1, -4  PRINT:  li $v0, 4  la $a0, mess  syscall  li $v0, 1  move $a0, $t0  syscall  END: li $v0, 10  syscall |
|  |

**3.4.**Write a MIPS program with the following requirements:

1. Declare an integer array with 10 synthetic data elements.

2. Print a sentence to terminal to request an integer number that is greater than 0 and less than 10 (assume that user strictly follow this rule).

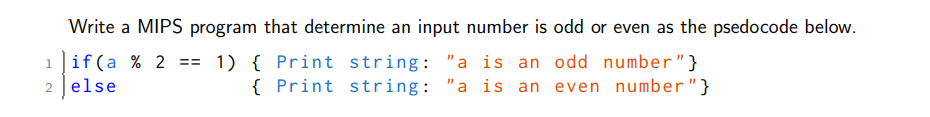
3. Print value of the element at index collected from the previous step.

|  |
| --- |
| .data  array: .word 0,1,2,3,4,5,6,7,8,9  mess: .asciiz "Input a number x (0 < x < 10): "  .text  li $v0,4  la $a0,mess  syscall    li $v0,5  syscall    move $t0,$v0  mul $t0,$t0, 4    la $s0,array  add $s0,$s0,$t0  lw $a0,0($s0)  li $v0,1  syscall |
|  |

**3.5.** Write a MIPS program that reverses an 10 elements integer array. For example, the array initially stores 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, the program will change the array to be 10, 9, 8, 7, 6, 5, 4, 3, 2, 1.

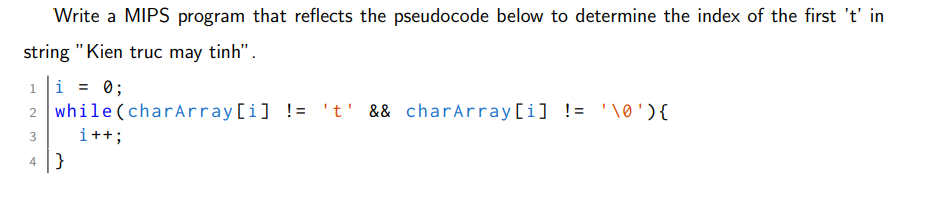
|  |  |
| --- | --- |
| .data  array: .word 1, 2, 3, 4, 5, 6, 7, 8, 9, 10  print: .asciiz "Print array: "  r\_print: .asciiz "\nAfter reversing: "  space: .asciiz " "  .text  la $a0, print  li $v0, 4  syscall    la $a1, array  jal printarr    la $a0, r\_print  li $v0, 4  syscall    jal r\_loop  jal printarr    li $v0, 10  syscall    printarr:  addi $t0, $0, 0  ploop:  beq $t0, 10, end  sll $t1, $t0, 2  add $t2, $t1, $a1    lw $t3, ($t2)  move $a0, $t3  li $v0, 1  syscall | la $a0, space  li $v0, 4  syscall    addi $t0, $t0, 1  j ploop    end: jr $ra    r\_loop:  addi $t0, $0, 0  rloop:  beq $t0, 5, r\_end  sll $t1, $t0, 2  add $t2, $t1, $a1  lw $t8, ($t2)    addi $t3, $0, 36  sub $t3, $t3, $t1  add $t4, $t3, $a1  lw $t9, ($t4)    sw $t8, ($t4)  sw $t9, ($t2)    addi $t0, $t0, 1  j rloop    r\_end: jr $ra |
|  | |

**3.6.**

****

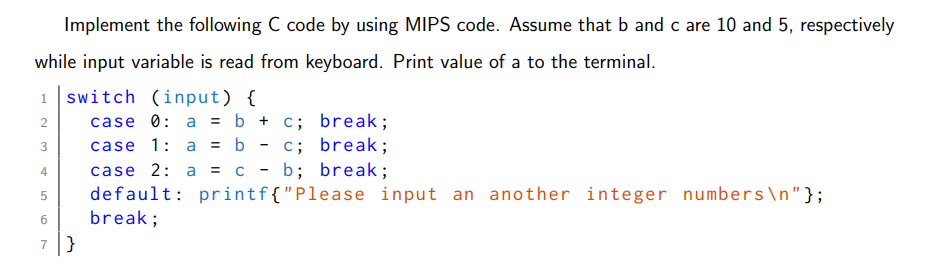
|  |  |
| --- | --- |
| .data  mess: .asciiz "Input number a: "  odd\_mess: .asciiz "a is an odd number"  even\_mess: .asciiz "a is an even number"  .text  li $v0, 4  la $a0, mess  syscall    li $v0, 5  syscall  move $t0, $v0    li $t1, 2  div $t0, $t1  mfhi $t1  beq $t1, 0,even | odd: li $v0,4  la $a0, odd\_mess  syscall    li $v0, 10  syscall    even: li $v0,4  la $a0, even\_mess  syscall    li $v0, 10  syscall |
|  | |

**3.7.**

****

|  |  |
| --- | --- |
| .data  char: .byte 't'  str\_s: .asciiz "Kien truc may tinh"  int\_t: .word 0  res\_mess: .asciiz "Index of the first 't' in string 'Kien truc may tinh': "  .text  .globl main  main:  lb $t1,char  la $t2,str\_s  lb $t3,0($t2)  addi $t0,$zero,0 #index = 0 | while:  beq $t3,$t1, end\_while  addi $t0,$t0,1  la $t2,1($t2)  lb $t3,0($t2)  j while  end\_while:  sw $t0,int\_t  print: la $a0,res\_mess  addi $v0,$zero,4  syscall  lw $a0,int\_t  addi $v0,$zero,1  syscall  end:  addiu $v0,$zero,10  syscall |
|  | |

**3.8.**

****

|  |  |
| --- | --- |
| .data  int\_a: .word 0  input: .word 13  mess1: .asciiz "Input: "  mess2: .asciiz "Please input an another integer numbers \n"  res\_mess: .asciiz "a = "  .text  .globl main  main:  la $a0,mess1  addi $v0,$zero,4  syscall    addi $v0,$zero,5  syscall  sw $v0,input  lw $t0,int\_a  lw $t1,input  addi $t2,$zero,0  addi $t3,$zero,0  add $t2,$t2,10  add $t3,$t3, 5 | #xet case  case:  beq $t1, 0, case1  beq $t1, 1, case2  beq $t1, 2, case3  j default  case1: add $t0,$t2,$t3  j endswitch  case2: sub $t0,$t2,$t3  j endswitch  case3: sub $t0,$t3,$t2  j endswitch  default:  la $a0, mess2  addi $v0,$zero,4  syscall  j end  endswitch: sw $t0,int\_a  print: la $a0,res\_mess  addi $v0,$zero,4  syscall  lw $a0,int\_a  addi $v0,$zero,1  syscall  end:  addiu $v0,$zero,10  syscall |
|  | |