StudentId_____

TE WHARE WĀNANGA O TE ŪPOKO O TE IKA A MĀUI
VICTORIA
UNIVERSITY OF WELLINGTON

# EXAMINATIONS — 2007

MID-YEAR

**COMP 203**

**COMPUTER ORGANISATION**

Time Allowed: 3 Hours (180 minutes)

Instructions: Answer all questions.
Make sure your answers are clear and to the point.
Calculators and paper foreign language dictionaries are allowed.
No reference material is allowed.
There are 180 possible marks on the exam.
Every box with a heavy outline requires an answer.
At the end of the script there is an appendix listing MIPS instructions and the assembly program of question 8.

| | Topic | Marks |
|---|---|---|
| PART I | | |
| 1 | Basic Concepts and Performance | *12 marks* |
| 2 | Machine Language and Buses/IO | *24 marks* |
| 3 | Computer Arithmetic | *24 marks* |
| PART II | | |
| 4 | Finite State machines | *10 marks* |
| 5 | Processor Data Path | *33 marks* |
| 6 | Pipelined Data Path | *25 marks* |
| 7 | Memory Hierarchy | *37 marks* |
| 8 | Disk and Input / Output | *15 marks* |

**Note:** Marks are shown for each question as a whole and also for their parts.

# PART I

## Question 1. Basic Concepts and Performance [12 marks]

**a) [4 marks]** Define each of the following terms **in one sentence** in the context of computer organisation:

**i)** Assembler

**ANSWER**

**ii)** Compiler

**ANSWER**

**iii)** Control

**ANSWER**

**iv)** Bit

**ANSWER**

**b) [3 marks]** For a given instruction set architecture, list **two possible ways** to increase the CPU performance in terms of the size of CPI and the clock rate.

**ANSWER**

**c) [5 marks]** Assume a computer A is used to execute a program P. The manufacturing manual shows that the computer CPU uses a 300MHz clock. A counting program C shows that program P has 150 billion instructions. When the program P is executed on computer A, the Unix command `time` gives the following results:

```
% time P (Return key)
1000u 250s 29:46 70%
```

Estimate the average CPI of the program P on computer A.

**ANSWER**

## Question 2. Machine Language [24 marks]

**a) [8 marks]** Consider the following segment of C code:

```
while (h < i)
{
     f = f + A[i];
     i = i + j;
}
f = f - j;
```

Assume that the registers $s0, $s1, $s2, $s3, and $s4 hold integer variables f, h, i, j, and the base address of integer array A, respectively. Insert a single instruction in each of the three outlined spaces labelled as 1, 2, 3 and 4, so that the resulting sequence of MIPS instructions directly corresponds to the above C code segment.

**ANSWER**

```
Loop: slt $t0, $s1, $s2

      1.

       add $t1, $s2, $s2
       add $t1, $t1, $t1

      2.

       lw $t2, 0($t1)
      add $s0, $s0, $t2
      add $s2, $s2, $s3

      3.

Exit:
      4.
```

# SPARE PAGE FOR EXTRA ANSWERS

Cross out the rough working that you do not want marked.
Specify the question number for work you do want marked.

**b) [8 marks]** Given the following C function:

```
int exam07(int a, int b, int c)
{
     int m;

     m = (a + b) + (c – 10);

     return m;
}
```

Assume that the registers `$a0, $a1` and `$a2` hold variables `a, b` and `c,` respectively, that register `$s2` is used to store the local variable `m`, and that both the caller and the callee need to use `$s2`. Insert a single instruction in each of the outlined spaces labelled as 1, 2, 3, and 4, so that the resulting sequence of MIPS instructions directly corresponds to the above procedure/function.

**ANSWER**

```
Exam07:
       addi $sp, $sp, -4

       1.

       add $t0, $a0, $a1
       addi $t1, $a2, -10
       add $s2, $t0, $t1

       2.

       lw $s2, 0($sp)

       3.

       4.
```

**c) [4 marks]** Consider the following MIPS instructions:

```
sltu $t0, $s1, $s2
slt $t1, $s1, $s2
```

Assume that

```
$s1 = 0110 0000 0000 0101 0000 0000 0000 1001
$s2 = 1000 0000 1001 0000 0000 0000 0000 1010
```

in binary, what will be the decimal values of `$t0` and `$t1`?

**ANSWER**

$t0 =

$t1 =

**d) [4 marks]** Consider the following sequence of MIPS instructions:

```
lui $s1, 0x02a5
addi $s2, $s1, 0x9453
ori $s3, $s1, 0x9453
```

What hexadecimal values will be stored in `$s1`, `$s2` and `$s3`?
(Hint: take care with the immediate operand)

**ANSWER**

$s1 =

$s2 =

$s3 =

## Question 3. Computer Arithmetic [24 marks]

As discussed in the lectures, bit patterns have no inherent meaning. They may represent signed integers, unsigned integers, floating point numbers, and even machine instructions. For questions a) to c), you may express your answers in the form like ($2^i$ +…+ $2^j$ + …+ $2^{-l}$ + …+ $2^{-k}$), where i, j, k, l∈{-31, -30,…,0, 1,..., 31}.

Given the bit pattern below:

```
1010 1101 0101 0000 0000 0010 0000 0000
```

What does it represent, if it is

**a) [3 marks]** A two's complement integer?

**ANSWER**

**b) [2 marks]** An unsigned integer?

**ANSWER**

**c) [4 marks]** A single precision floating point number (Assuming the above bit pattern is in IEEE 754 binary representation)?

**ANSWER**

**d) [5 marks]** A MIPS instruction?

**ANSWER**

**e) [10 marks]** This question concerns overflow detection and manipulation.

Suppose that A and B are two negative integers stored in registers $s1 and $s2, respectively. Write a sequence of MIPS instructions (at most 10) to process all the following tasks:

- perform `C = A + B;` (Store C in register $s3)
- if there is no overflow, add constant 50 to C and place the result in register $s4;
- otherwise, set the most significant bit of C to 1, and store a carry in `$s5.`

Use temporary registers if necessary.

**ANSWER**

# SPARE PAGE FOR EXTRA ANSWERS

Cross out the rough working that you do not want marked.
Specify the question number for work you do want marked.

# PART II

## Question 4. Finite State Machines [10 marks]

A flip flop is a (primitive) Finite State Machine (FSM).

**a) [5 marks]** Describe the operation of a flip flop using the State Transition Table provided below.

**ANSWER**

| Current State (Q) | Input (D) | Next State (Q') |
|---|---|---|
| | | |
| | | |
| | | |
| | | |

**b) [5 marks]** Use a State Transition Diagram to describe the operation of a flip flop graphically.

**ANSWER**

## Question 5. Processor Data Path [33 marks]

**a) [2 marks]** What is the program counter in the MIPS processor architecture and what is it used for?

**ANSWER**

**b) [4 marks]** What is a register file in the MIPS processor architecture and what is it used for?

**ANSWER**

**c) [4 marks]** What is a sign extender in the MIPS processor architecture and what is it used for?

**ANSWER**

**d) [4 marks]** What is the role of a left shifter for two binary position in the MIPS processor architecture?

**ANSWER**

**e) [5 marks]** Give the names of 5 cycles of the MIPS Multi Cycle Data Path.

**ANSWER**

| |
|---|
| |

**f) [8 marks]** Fill in the table below to show the main differences between MIPS Single Cycle Data Path and Multi Cycle Data Path structures (Memories, ALU's, Multiplexers, Registers).

**ANSWER**

| Single Cycle Data Path | Multi Cycle Data Path |
|---|---|
| | |
| | |
| | |
| | |
| | |
| | |

**g) [6 marks]** Consider the diagram in Figure 5. On the diagram, show only those lines and components that will be used to execute the fourth cycle of a MIPS **add** instruction. Trace the lines and circle the components using a coloured pen.

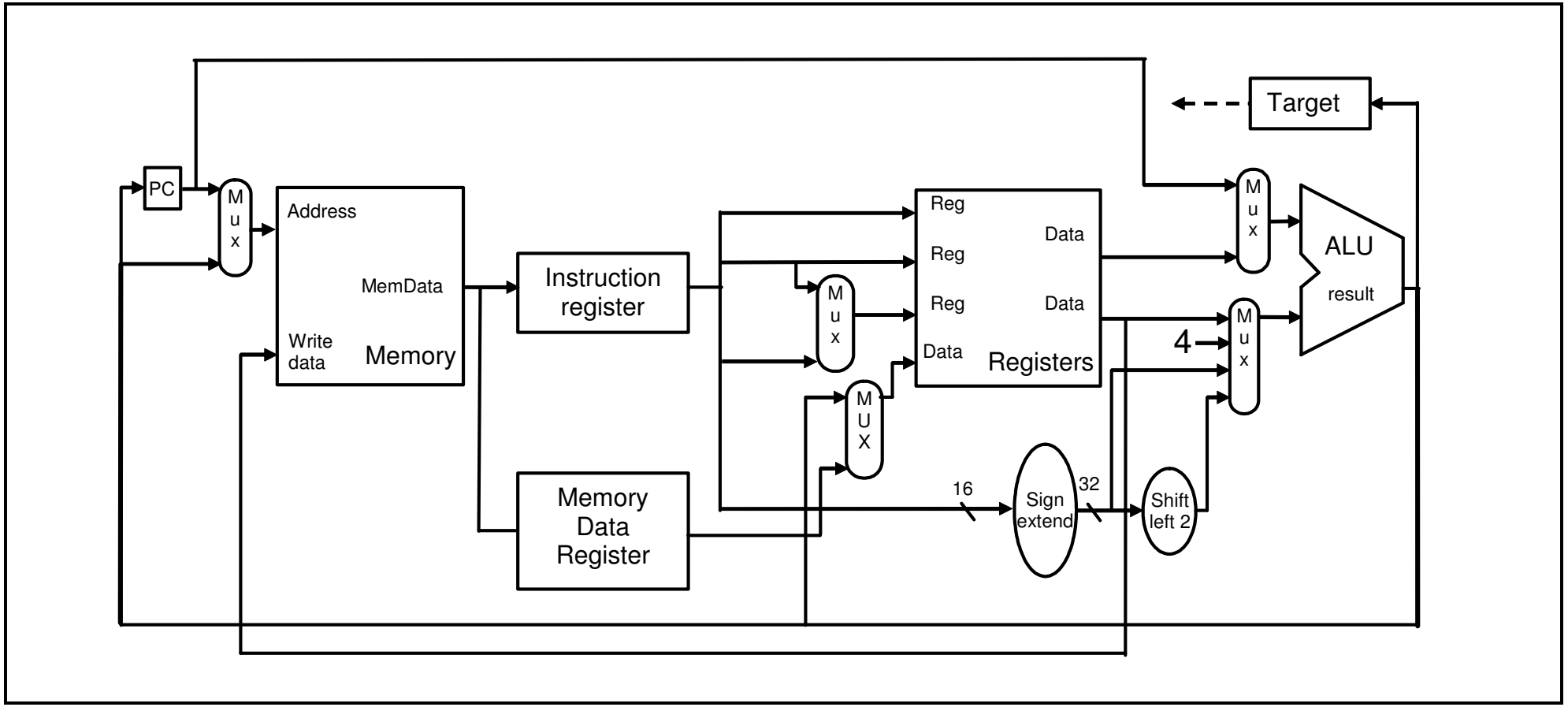


**Figure 5** Fourth Cycle of a MIPS **add** instruction

# Question 6: Pipelined Data Path [25 marks]

**a) [4 marks]** What is pipelining and what is the expected benefit of pipelining?

**ANSWER**

**b) [4 marks]** Hazards are the main problem of pipelined processors. Name the two most common kinds of hazards.

**ANSWER**

**c) [5 marks]** List the techniques that are used to fight against hazards?

**ANSWER**

**d) [12 marks]** Consider the MIPS assembly program given below. The assembler supports branch labels. At the start of the program, all processor registers are set to zero.

```
addi $1, $0, 2
sw $1, 0($0)
add $1, $0, $0
addi $2, $0, 10
loop:
lw $3, 0($0)
slt $4, $2, $3
bne $0, $4, stop #branch to stop label to terminate the program
addi $1, $1, 1
addi $5, $3, 2
sw $5, 0($0)
beq $0, $0, loop # branch to loop label
stop:
end #terminate the execution
```

**i. [4 marks]** Suppose the program above is executed in a pipelined data path with forwarding. The decision whether to branch or not is made in the second stage, and if a branch is taken the PC is also updated in the second stage. Branch Assumption is set to off. What will be the content of the register $1 if we execute the program as it is?

**ANSWER**

**ii.** **[8 marks]** Insert NOPs in only those places in the program above that are needed for its correct execution in a pipelined data path with:

- Forwarding,
- Where the decision whether to branch or not is made in the second stage, and
- If a branch is taken the PC is updated also in the second stage.

Show your working in the space provided for the answer below.

**ANSWER**

## Question 7. Memory Hierarchy [37 marks]

**a) [2.5 marks]** List the names of the memory hierarchy levels.

**ANSWER**

**b) [2 marks]** What are the two phenomena that the memory hierarchy is based on?

**ANSWER**

**c) [3 marks]** What is a cache memory and what is it used for?

**ANSWER**

**d) [4 marks]** What is a virtual memory, what is it used for, and what is the name of a computer program that manages the virtual memory?

**ANSWER**

**e) [3 marks]** What is a page table used for and where is it kept?

**ANSWER**

**f) [3 marks]** What is a Translation Lookaside Buffer (TLB) and what is it used for?

**ANSWER**

**g) [19.5 marks]** Consider the assembly program given below.

```
#This program is constructed to test your understanding of set
#associative caches and has no other purpose.
#All registers of the Register File are initially set to 0.

#Data declaration section sets main memory
#words 16 and 20 to values 10 and 20, respectively.
.data
.word 0, 0, 0, 0, 10, 20
.text

#Here starts the code
addi $3, $0, 3
addi $4, $0, 4
addi $6, $0, 2 #number of loops
loop:
addi $5, $5, 1 #i++
lw $1, 16($0)
lw $2, 20($0)
mul $1, $1, $3
sw $1, 32($0)
mul $2, $2, $4
sw $2, 36($0)
addi $3, $3, 1
addi $4, $4, 1
bne $5, $6, loop
end
```

**Note**: There is a copy of the assembly program above given at the end of the appendix. Tear it off and use to answer the question.

Suppose the program above is executed in a MIPS processor. The processor has a cache with the following properties:

- Cache size is 8W
- Cache block size is 2W
- Cache type is 2-way set associative,
- Replacement algorithm is FIFO, and
- Update policy is Write-Back.

The cache - memory system has the following performance figures:

- Cache read time: 1 processor cycle,
- Cache write time: 1 processor cycle,
- Time to read a two word block from main memory: 12 processor cycles,
- Time to write a two word block to main memory: 12 processor cycles.

The program executes the loop two times and terminates successfully.

**i) [8 marks]** Show the content of the cache after the first pass through the program loop in the cache layout provided below. (Show contents of tag fields as hexadecimal numbers, show contents of of Word 0 and Word 1 as decimal numbers.)

**ANSWER**

| Slot No | Set No | Valid | Dirty Bit | Tag | Word 0 | Word 1 |
|---|---|---|---|---|---|---|
| 0 | 0 | | | | | |
| 1 | 0 | | | | | |
| 2 | 1 | | | | | |
| 3 | 1 | | | | | |

**ii) [5 marks]** What are:

- The number of read hits,
- The number of read misses,
- The number of write hits,
- The number of write misses, and
- The miss penalty (in the number of processor cycles)

after the first pass?

**ANSWER**

Read Hits =

Read Misses =

Write Hits=

Write Misses =

Miss Penalty =

# SPARE PAGE FOR EXTRA ANSWERS

Cross out the rough working that you do not want marked.
Specify the question number for work you do want marked.

**iii) [4 marks]** Show the content of the cache after the second pass through the program loop in the cache layout provided below.

**ANSWER**

| Slot No | Set No | Valid | Dirty Bit | Tag | Word 0 | Word 1 |
|---|---|---|---|---|---|---|
| 0 | 0 | | | | | |
| 1 | 0 | | | | | |
| 2 | 1 | | | | | |
| 3 | 1 | | | | | |

**iv) [2.5 marks]** What is:

- The total number of read hits,
- The total number of read misses,
- The total number of write hits,
- The total number of write misses, and
- The total miss penalty

after the termination of the program?

**ANSWER**

Read Hits =

Read Misses =

Write Hits=

Write Misses =

Miss Penalty =

# SPARE PAGE FOR EXTRA ANSWERS

Cross out the rough working that you do not want marked.
Specify the question number for work you do want marked.

## Question 8. Disk and Input / Output [15 marks]

A program repeatedly performs a three-step process:

- It reads in a 4-KB block from a disk into the main memory,
- Does some processing of that data, and then
- Writes back as a 4-KB block in the location with the same address on the disk.

Each block is contiguous and randomly located on a single track on the disk. The disk drive:

- Rotates at 10000 rotations per minute,
- Has an average seek time of 4 ms, and
- Has a transfer rate of 16 MB/sec.
- The disk controller overhead is negligible.

No other program is using the disk or processor, and there is no overlapping of disk operation with processing.

The processing step takes 500,000 clock cycles, and the processor clock rate is 1000MHz.

The disk is controlled by a Direct Memory Access (DMA) disk controller. To initiate a disk operation, the processor uses 1000 cycles. To check correctness of a finished disk operation, processor uses 500 cycles.

**a) [7 marks]** Calculate the average time to read a block of data from disk and transfer it to the main memory.

**ANSWER**

**b) [3 marks]** Calculate the average time to write back a block of data on the disk unit.

**ANSWER**

**c) [5 marks]** What is the average throughput of the system in blocks read, processed, and written back per second?

**ANSWER**

**********

# APPENDIX

## Commonly Used MIPS Instructions

### Arithmetic and Logical Instructions

**add** `Rdest, Rsrc1, Src2`

Addition (with overflow)

**addi** `Rdest, Rsrc1, Imm`

Addition Immediate (with overflow)

**addu** `Rdest, Rsrc1, Src2`

Addition (without overflow)

**and** `Rdest, Rsrc1, Src2`

AND

**andi** `Rdest, Rsrc1, Imm`

AND Immediate. Put the logical AND of the integers from register Rsrc1 and Src2 (or Imm) into register Rdest.

**or** `Rdest, Rsrc1, Src2`

OR

**ori** `Rdest, Rsrc1, Imm`

OR Immediate. Put the logical OR of the integers from register Rsrc1 and Src2 (or Imm) into register Rdest.

**sll** `Rdest, Rsrc1, Src2`

Shift Left Logical

**srl** `Rdest, Rsrc1, Src2`

Shift Right Logical

**sub** `Rdest, Rsrc1, Src2`

Subtract (with overflow)

**subu** `Rdest, Rsrc1, Src2`

Subtract (without overflow) Put the difference of the integers from register Rsrc1 and Src2 into register Rdest.

### Constant-Manipulating Instructions

**li** `Rdest, imm`

Load Immediate. Move the immediate imm into register Rdest.

**lui** `Rdest, imm`

Load Upper Immediate Load the lower halfword of the immediate imm into the upper halfword of register Rdest. The lower bits of the register are set to 0.

## Comparison Instructions

**slt** Rdest, Rsrc1, Src2
: Set Less Than

**slti** Rdest, Rsrc1, Imm
: Set Less Than Immediate

**sltu** Rdest, Rsrc1, Src2
: Set Less Than Unsigned

## Branch and Jump Instructions

**beq** Rsrc1, Src2, label
: Branch on Equal. Conditionally branch to the instruction at the label if the contents of register Rsrc1 equals Src2.

**bne** Rsrc1, Src2, label
: Branch on Not Equal. Conditionally branch to the instruction at the label if the contents of register Rsrc1 are not equal to Src2.

**j** label
: Jump. Unconditionally jump to the instruction at the label.

**jal** label
: Jump and Link

**jalr** Rsrc
: Jump and Link Register. Unconditionally jump to the instruction at the label or whose address is in register Rsrc. Save the address of the next instruction in register 31.

**jr** Rsrc
: Jump Register. Unconditionally jump to the instruction whose address is in register Rsrc.

## Load Instructions

**lw** Rdest, address
: Load Word. Load the 32-bit quantity (word) at *address* into register Rdest.

**lb** Rdest, address
: Load Byte. Load the 8-bit quantity (byte) at *address* into register Rdest.

## Store Instructions

**sw** Rsrc, address
: Store Word. Store the word from register Rsrc at *address*.

**sb** Rsrc, address
: Store Byte. Store the byte from register Rsrc at *address*.

# Commonly Used MIPS Fields

There are six commonly used MIPS fields: *op, rs, rt, rd, shamt*, and *funct*. The *op* and *funct* are usually used to represent and distinguish between different operations/instructions. The following table gives the *op* and *funct* for the commonly used MIPS instructions.

| **Instructions** | **op** | **Funct** |
|---|---|---|
| add | 0 | 32 |
| sub | 0 | 34 |
| lw | 35 | NA |
| sw | 43 | NA |
| beq | 4 | NA |
| bne | 5 | NA |
| slt | 0 | 42 |
| j | 2 | NA |
| jal | 3 | NA |
| jr | 0 | 8 |

Registers:

| Name | Number | Usage |
|---|---|---|
| $zero | 0 | constant value 0 |
| $at | 1 | reserved for assembler |
| $v0–$v1 | 2–3 | values for results and expression evaluation |
| $a0–$a3 | 4-7 | arguments, for functions/procedures |
| $t0–$t7 | 8-15 | temporaries |
| $s0–$s7 | 16-23 | saved. Fast locations for data |
| $t8–$t9 | 24-25 | more temporaries |
| $k0–$k1 | 26-27 | reserved for the OS |
| $gp | 28 | global pointer |
| $sp | 29 | stack pointer |
| $fp | 30 | frame pointer |
| $ra | 31 | return address, for functions/procedures |

Question 8.g assembly program

```
#This program is constructed to test your understanding of set
#associative caches and has no other purpose.
#All registers of the Register File are initially set to 0.

#Data declaration section sets main memory
#words 16 and 20 to values 10 and 20, respectively.
.data
.word 0, 0, 0, 0, 10, 20
.text

#Here starts the code
addi $3, $0, 3
addi $4, $0, 4
addi $6, $0, 2 #number of loops
loop:
addi $5, $5, 1 #i++
lw $1, 16($0)
lw $2, 20($0)
mul $1, $1, $3
sw $1, 32($0)
mul $2, $2, $4
sw $2, 36($0)
addi $3, $3, 1
addi $4, $4, 1
bne $5, $6, loop
end
```