National Exams December 2016

98-Comp-A3, Computer Architecture

3 hours duration

NOTES:

1. If doubt exists as to the interpretation of any question, the candidate is urged to submit with the answer paper, a clear statement of any assumptions made.

2. This is a CLOSED BOOK EXAM.
   Any non-communicating calculator is permitted.

3. FIVE (5) questions constitute a complete exam paper.
   The first five questions as they appear in the answer book will be marked.

4. Answering questions 1 and 2 is mandatory.

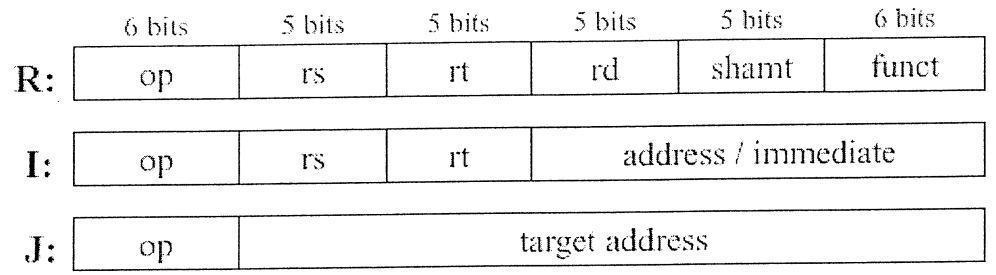
5. The questions are not of equal value.

6. Some questions require an answer in essay format. Clarity and organization of the answer are important.

Marking Scheme

1. (a) 10 marks (b) 5 marks (c) 5 marks
2. (a) 5 marks (b) 5 marks (c) 5 marks (d) 5 marks
3. (a) 5 marks (b) 5 marks (c) 5 marks
4. (a) 5 marks (b) 10 marks
5. (a) 5 marks (b) 10 marks
6. 15 marks

1. **(a)** The MIPS32 instruction set uses 32 general purpose integer registers and 32 floating point registers. An instruction can have up to one destination register and up to two source registers. The following three instruction formats exist, where only the "R" and "I" format can reference registers as explained below (rt is a destination register for the "I" format):



op: basic operation of the instruction (opcode)
rs: first source operand register
rt: second source operand register
rd: destination operand register
shamt: shift amount
funct: selects the specific variant of the opcode (function code)
address: offset for load/store instructions (+/-2^{15})
immediate: constants for immediate instructions

(i) If we were to add more registers to MIPS what would be the potential benefits? What would be the challenges to adding more registers to MIPS?

(ii) Discuss the benefits of having separate integer and floating point register files.

(b) A processor executes a sequence of 20 instructions out of which only three are memory reads and one is a memory write. Each of these memory reads or writes accesses 2 bytes of memory data. Each instruction is encoded using 16 bits. What is the minimum number of bytes that a processor will have to read from memory when it executes these 20 instructions?

(c) A processor's memory contains the value 0x0012 6700 at address 0x8012 0234. What can this value be? An instruction? Data? Either?

2. **(a)** Two 8-bit registers A and B contain the values 0xFE (shown in hexadecimal) and 0x10, respectively. What is the value of A + B in **decimal**, if (i) A and B were to be interpreted as unsigned integers, and (ii)

if A and B were to be interpreted as signed integers in 2's complement. Explain your answer.

(b) If A and B are unsigned integers represented using 32 bits, can A + B always be represented in the same representation? Explain your answer with an example if appropriate.

(c) The IEEE standard for single precision floating point representation of real numbers uses 32-bits in total comprising an 8-bit exponent E, a sign bit S, and a 23 bit mantissa M. The number encoded is:

(-1)^S x 2^(E – 128) x 1.M

Given three floating point numbers A, B, and C does the following equation hold? (A + B) + C = A + (B + C). Explain your answer.

(d) Explain how a unidimensional array A of 64 32-bit values will be stored in memory. Where in memory would be A[i] if the array is stored starting from memory address 0x1000?

3. **(a)** A processor has a 4GB byte-addressable address space. How will a 128KB, 4-way set-associative cache with 64-byte blocks be indexed. Explain your answer.

(b) A block cached in set 0x10 (hexadecimal) of the cache described in part (a) is tagged with 0x01 (hexadecimal). What is the range of addresses it contains in hexadecimal?

(c) A processor performs a sequence of 10 memory reads each reading a byte from memory. The processor has a data and an instruction cache, each using 32 byte blocks. Assume that all instruction fetch accesses are hits in the instruction cache and that the data cache is empty. On a miss, the data cache fetches a complete 32-byte block from memory. What is the minimum number of bytes that the data cache will have to read from memory to service all 10 memory reads? What is the maximum?

4. **(a)** In the following assembly code identify all RAW (read-after-write), WAR (write-after-read) and WAW (write-after-write) dependencies:

```
L0:    addi r9, r0, 0x500   # r9 = r0 + 0x500 (hex)
L1:    addi r8, r0, 0x400   # r8 = r0 + 0x400
L2:    ldw r7, 0(r8)        # r7 = Memory[r8 + 0]
L3:    stw r7, 0x800(r8)    # Memory[r8 + 0x800] = r8
L4:    addi r8, r8, 4       # r8 = r8 + 4
L5:    bne  r8, r9, L2      # if (r8 == r9) PC = L2
```

R0-R31 are registers. Mark dependencies as (x,y,rz) pairs where x and y code labels and where y depends on x through register rz. Make sure to specify the dependency type, e.g., RAW (L100,L200, R31)

(b) You are the chief architect of the controller processor of an autonomous snowplow. To keep costs in check, the chip area can be up to 4 mm^2 (for example, it could be 2mm x 2mm square). The controller software operates in phases during which it processes sensor data such as video and adjusts the snowplows actuators (e.g., speed and direction) accordingly. One option would be to use a powerful pipelined core (PC) which takes 100ms to process each phase. While 20% of those 100ms are sequential, the remaining 80% can be parallelized with no overhead.

You have three processing elements which you can place on the 2mm^2 die at any combination you wish:

1. A powerful pipelined core (PC) that needs 4 mm^2 of area and executes each processing phase exactly in 100ms.
2. Simple pipelined cores (SC), each needing 1 mm^2 of area. Each executes code at half the speed of PC.
3. Accelerator cores (AC), each needing 1.5 mm^2 of area, and that execute code at 1.5x the speed of PC.

A working controller needs to have at least one PC or SC to run the sequential part of the workload. The AC cores can only be used for the parallelizable portion of the workload. All three cores can be shaped as desired to make use of the chip area. For example, an SC can be shaped as a 1mm x 1mm square, or as a 0.5mm x 2mm rectangle, or any other shape that occupies 1mm^2. Which configuration would achieve the shortest execution time per phase? Demonstrate your answer by calculating the execution time for all valid configurations.

5. **(a)** A memory chip has the following interface: A0-A15 are 16 single bit input address lines specifying which row is accessed, a single bit input signal R/W! specifies whether the access is a read (1) or a write (0), E is a single bit input signal that must be 1 to access the chip. The data values that are read or written appear on the four D3 through D0 single bit output/input pins. When E is 0 the D3-D0 pins are in high-Z. What is the total capacity of this memory chip in bytes?

(b) Using as many as necessary of the chips described in part (a), synthesize the equivalent of an 8-bit wide memory that has a 32KB total capacity. You can use a few additional logic gates as needed.

6. A 32-bit unsigned integer **cnt** is stored at memory address 0x1000 and initialized to **10**. The following assembly pseudo code shows an interrupt handler and a main program. The main program continuously increments cnt while the interrupt handler decrements it using the instructions shown. Interrupts can occur between any two instructions of the main program. In response to an interrupt the processor disables further interrupt requests and executes the interrupt handler to completion. Upon return from the interrupt handler execution resumes where it was interrupted and interrupts are enabled again:

```
Interrupt Handler:                  # interrupts disabled
I0:       movia r8, 0x1000          # r8 = address of cnt
I1:       ldw r9, 0(r8)             # r9 = mem[0x1000]
                                    # read cnt from memory
I2:       addi r9, r9, -1           # r9 = r9 - 1
I3:       stw r9, 0(r8)             # write new value to
                                    #cnt in memory
          eret                      # return to main program
                                    # interrupts are enabled
main:
          movia r8, 0x1000 # r8 = address of cnt
wait:
W1:       ldw r11, 0(r8)   # read cnt from memory
                           # into register r11
W2:       add r11, r11, 1  # r11 = r11 + 1
W3:       stw r1, 0(r8)    # store r11 into cnt
W4:       br wait          # back to W1/wait:
```

Assume that interrupts can happen at any time requested by external devices. If we could observe the sequence of writes to cnt done by the stores at I3 and W3, which of the following value sequences are possible? If so, explain why by showing the sequence of instructions that get executed (e.g., W1, W2, W3, W4, I1, I2, I3, ..., etc.; if not, give a brief explanation as to why not): **(i) 11, 12, 13, and (ii) 11, 9, 12.**