Name: Bhaskaran Raman Roll number: Sample Answers

CS230 DLDCA End-Sem, Tue 12 Nov 2024, 13.30-16.30, Max. Marks: 40

General instructions

- Write only in the space provided. Answer briefly but crisply (not lengthily or loosely).
- You are allowed to refer to your own hand-written notes only.
- Write neatly and clearly. Up to **+2 HP** for neat handwriting, neat/crisp answers.
- Answers generally have to be (briefly) explained. State any necessary assumptions.

[Q1] Short answer questions: [1x10=10 marks]

1. What real instruction(s) does the pseudo-instruction 'la' (load address) translate to?

   la load the address of an assembly variable onto a register
   It translates to:
   lui <reg>, <upper-16-bits> # not needed if constant value is <= 16 bits
   ori <same reg>, <same reg>, <lower-16-bits> # can also use addi here

2. MIPS has a convention as to which registers are caller saved versus which are callee saved. What is the need for such a convention?

   Such a convention is needed to link independently written/complied code, e.g. 3rd party library

3. Are RAW hazards in memory locations possible in the 5-stage MIPS pipeline? Explain.

   All memory operations in 4th stage (MEM), which always happens in order of instruction execution. So no RAW hazards in mem locations in 5-stage MIPS pipeline

4. State two reasons why cache memory is faster than main memory.

   Any two of: smaller, SRAM instead of DRAM, closer to processor

5. A 32-bit machine has 1GB of physical memory and a 4KB page size. What is the maximum index of an integer array a program can use in this machine? Assume sizeof(int) is 1 word.

   Max index governed by VA space = floor[(2^32-1)/4] = 2^30-1

6. State one use of virtual memory as a level of indirection.
   Any one of: illusion of memory being larger than physical memory, program relocation is easy without program rewrite, protection of one process's memory from another, controlled sharing of memory locations across processes, copy-on-write for shared dynamic linked library

7. Some machines have a TLB with a PID (process-ID) field. What is the purpose behind this?

   To avoid requirement of TLB flush on process switch (context switch)

8. State one advantage of having a long pipeline, i.e. a large number of stages

   Pipeline speedup is proportional to pipeline length – more parallelism

9. State one disadvantage of having a long pipeline, i.e. a large number of stages

   Any one of: control is complex, exception handling is complex, data forwarding logic is complex, more chance of dependence across instructions in pipeline

10. What are the three kinds of bus lines?

    Control, address, data lines

[Q2] Majority Gate [1+1+1+2=5 marks]

Consider the following function: M (x, y, z) = xy + yz + xz. This is called a *majority* function/gate. This is because the function evaluates to 1(0) only if at least two of the 3 variables are 1(0). In this question, we shall be exploring the majority gates. Turns out that such majority functions have very interesting applications in nanotechnology-based circuits. Implement the following functions using only majority gates. You may use 0 or 1 as input where appropriate. You may also assume that where needed, the complemented input x' of an input variable x is also available, in addition to x itself.

1. F(x) = x ; use exactly one majority gate

   F(x) = M(x, 1, 0)

2. F(x,y) = x+y ; use exactly one majority gate

   F(x,y) = M(x, y, 1)

3. F(x,y) = xy ; use exactly one majority gate

   F(x, y) = M(x, y, 0)

4. F(x,y,z) = xy' + y'z ; use exactly two majority gates

   F(x,y,z) = (x+z)y' = M(M(x,z,1), y', 0)

[Q3] MIPS ISA [5 marks]

Consider the conditional move instruction MOVZ which copies a source register to a destination register only if a third register is zero. For example, the instruction

MOVZ $8, $11, $4

copies the contents of register 11 into register 8, only-if register 4 is zero (otherwise it does nothing).

1. **[1 mark]** Machine M1 is a regular MIPS machine whose assembler implements MOVZ as a pseudo-instruction. Indicate how MOVZ $a, $b, $c can be implemented using real MIPS instruction(s).

```
        bne $c, $0, SKIP
        add $a, $b, $0 # Could also be addi $a, $b, 0
SKIP:
```

2. **[1 mark]** Your answer above would have used a branch instruction. Give the 16-bit immediate value in the machine code of this branch instruction.

   16-bit immediate value = word offset from PC+4
   SKIP is at PC+8 with respect to the branch
   So answer = [(PC+8)-(PC+4)]/4 = 1

3. **[2 marks]** Machine M2 extends the MIPS ISA by implementing MOVZ as a real instruction. A program P while executing on M2, has 5% of executed instructions as MOVZ. For this program, M1 and M2 have the same MIPS (Millions of Instructions Per Second) rating and the same CPI. Which machine executes P faster and by what factor?

   Time/Program = [Instructions/Program] / [Instructions/Time]
   M1 and M2 have the same MIPS (Millions of Instructions Per Second) => same Instructions/Time
   M1 has 1.05 times more executed instructions as M2, i.e. 1.05 times more Instructions/Program
   So M2 is faster by a factor of 1.05

4. **[1 mark]** M2's clock speed is 2.1GHz. What is M1's clock speed?
   Instructions/Time = [Instructions/Cycle] * [Cycle/Time]
   = [1/CPI] * [Cycle/Time] = [1/CPI] * clockSpeed
   M1 and M2 have same Instructions/Time and same CPI, so they must have same clockSpeed
   Ans: 2.1GHz

[Q4] Pipelining [11 marks]

Consider the same MOVZ instruction as above (except for the definition of MOVZ, this question is independent of the previous question). Machine M2 implements MOVZ by minimally modifying the original 5-stage MIPS pipeline implementation.

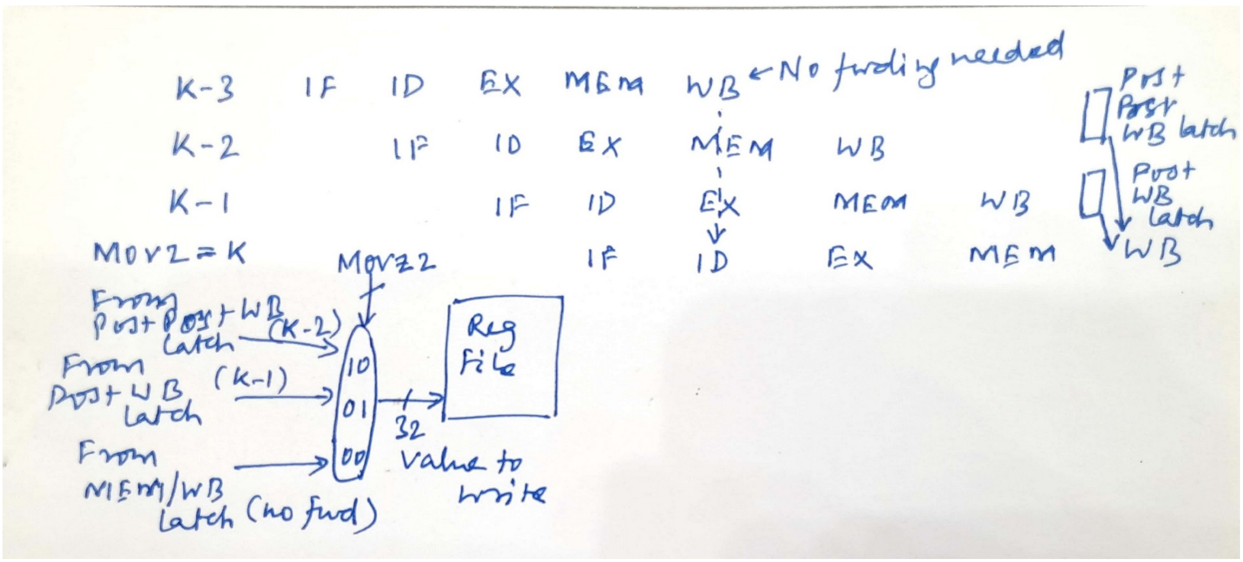
1. **[2 marks]** Suggest an appropriate instruction format for MOVZ \$a, \$b. \$c. Provide a drawing with justification.
   3-regs, so R-format is suitable; \$a should be Rd; \$c is compared with 0, so it should be Rs for minimal changes to datapath (show the drawing)
   Giving \$b as Rs and \$c as Rt is also acceptable

2. **[1 mark]** In which stage of execution does MOVZ need the RegWr (Register Write) control line? Explain briefly.
   It needs it in the WB stage, i.e. 5th stage

3. **[1 marks]** At the end of which stage of MOVZ's execution can the RegWr line be determined?

   Assuming same ALU is used for comparing \$c to 0, the result of comparison is known at the end of the 3rd cycle, i.e. EX cycle; thus RegWr line is known at the end of the EX cycle

4. **[3 marks]** Consider data forwarding for the MOVZ instruction when the source register of the 'move' has an RAW dependence on a previous instruction (consider *only* these instances of data forwarding). Draw an appropriate pipeline timing diagram to illustrate the possible scenarios of data forwarding. Draw also the datapath modifications necessary to support these scenarios of data forwarding.



5. **[3 marks]** Write the pseudo-micro-code logic to generate the control line(s) required for the above data forwarding. The logic should be written to execute in the dependent MOVZ's ID stage. You need to consider *only* register-register instructions as possible data producing instructions.
   (Answer assumes src reg $b is Rt – could be Rs – check consistency with answer to Q5.1 – also check specific values of control lines from Q5.4)

```
MOVZ2 = 00 // No forwarding is default
if ( (IF/ID.funct == MOVZ) && (IF/ID.opcode == 0) &&
    (IF/ID.Rt == ID/EX.Rd) && ID/EX.RegWr && !(ID/EX.MemRd) )
        MOVZ2 = 01 // Fwd from K-1
else if ( (IF/ID.funct == MOVZ) && (IF/ID.opcode == 0) &&
    (IF/ID.Rt == EX/MEM.Rd) && EX/MEM.RegWr && !(EX/MEM.MemRd) )
        MOVZ2 = 10 // Fwd from K-2
```

   Note: logic should work when there is dependence on *both* K-2 and K-1, in which case K-1 should take precedence

[Q5] Delay slot scheduling [4 marks]

Consider the same MOVZ instruction as above (except for the definition of MOVZ, this question is independent of the previous questions). Machine M2 implements MOVZ by extending the original 5-stage MIPS pipeline implementation. It has a 2-stage branch completion scheme. It also has a branch delay slot. Construct an example assembly language code snippet (a simple one) where MOVZ helps fill a branch delay slot which cannot be filled had there been no MOVZ in the instruction set.

```
        addi R1, R1, 1
        beq R1, $0, SKIP
        nop # delay slot
        sw R3, 4($sp)
SKIP:
        mov R3, R2 # add R1, R2, $0
```

Note that we cannot schedule the addi in the delay slot since the beq is data dependent on the addi
We cannot schedule the sw in the delay slot as it modifies a memory location
Without MOVZ, we cannot schedule the mov in the delay slot, as it would change the value stored by sw
With MOVZ, we can rewrite the mov as: MOVZ R3, R2, R1 and schedule it in the delay slot safely – it will have no effect on the sw if R1 happens to be non-zero

[Q6] Paged page tables: 1+1+3=5 marks

A computer has a 32-bit VA space, and 64GB of main memory. Page table entries are of size 1 word, and the page size is 64KB. It uses a paged (or virtualized) page-table scheme. Answer the following questions.

1. What is the per-process page-table size (in VM) in this scheme ?
   VA = 32 bit, page size = 64KB = 16 bits
   Num virt pages = 2^32/2^16 = 2^16
   PT size = 2^16 x 1 word = 2^16 words = 2^18 bytes
2. The OS stores page tables in a special process's VM, starting from virtual address 0x1000 0000. The page-table of process with PID *p* is in virtual address 0x1000 0000 + *p* * (PT size). Now, a process with PID 256 is running. What is the value of the page-table register (in hex format) ?
   PT reg = starting location of PT = 0x 1000 0000 + 256 * 2^18 = 0x 1400 0000
3. The page-table of the special OS process is pinned, starting from PA 0x 0 8000 0000. The process with PID 256 accesses VA 0x 0002 A804, which results in a TLB miss. What is the first physical address (express in hex format) accessed while handling this TLB miss? Assume all caches/TLB to be cold to begin with.
   VA = 0x 0002 A804 => Virt page num = 0x 0002 = 2
   PT starts in VM of OS at 0x 1000 0000 + 256 * 2^18 = 0x 1400 0000
   PT entry is at virt addr = 0x 1400 0000 + 2*4 = 0x 1400 0008
   Virt page num of this virt addr = 0x 1400
   Its PT entry in OS's PT will be at 0x 8000 0000 + 0x1400 * 4 = 0x 8000 5000
   This is the first physical address accessed

[Q7] Dharavi and the "once-in-a-century" "pandemic" [optional, 10HP]

The official claim regarding Covid is that it was a highly transmissible, deadly, "once-in-a-century" "pandemic", overwhelming hospitals everywhere. The Dharavi slum is one of the densest and poorest places on earth, with poor access to healthcare. What percentage of people in Dharavi died of Covid? (The next time you meet someone from the working class, ask them whether they found the "pandemic" deadly as claimed).
About 350 Covid deaths among about 800,000 estimated population over 2 years = 0.04% = 0.02% per year
It strains credulity to call this a pandemic overwhelming hospitals
Hospitals were overwhelmed (and people died) in Delhi, New York, London, etc due to exaggerate panic and quack protocols of distancing and killing people with fear itself