Name: _____

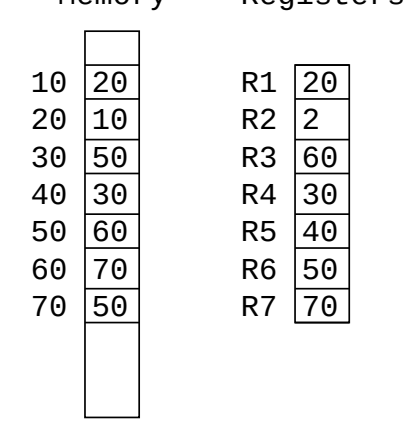
Practice Test 1

Updated: September 30, 2019

Test 1 is (tentatively) Monday 14 October.

1. List and explain several ways of measuring performance. Give advantages and disadvantages of each.
2. Explain the difference between latency and throughput.
3. What is the main risk in evaluating performance using throughput alone?
4. What is an *accumulator*?'
5. How has changing technology affected the way we design CPUs?
6. What is a benchmark?
7. How are benchmarks used?
8. Describe one or more alternatives to benchmarks.
9. How do benchmarks combine the results of multiple programs into a single number? Why is this technique used?
10. Why is it often difficult to say with certainty that one CPU performs better than another?
11. What is a *word* as it relates to CPU design?
12. List and explain the direct and indirect effects of a computer's word size (16-bit, 32-bit, 64-bit, etc.).
13. Compute the value of R1 at the end of each of the following instructions. The given addressing mode applies to the *third* parameter only. `R1` and `R2` are always register direct.

Instruction	Addressing mode	Result
ADD R1,R2, 40	memory indirect	
ADD R1,R2, 40	immediate	
ADD R1,R2, 40	memory direct	
ADD R1, R2, R3	register indirect	
ADD R1, R2, R3	register direct	



Name: _____

14. For each of the addressing modes above, describe a specific situation where instructions using that addressing mode would be useful. Use ideas and examples from a high-level programming language to clarify your explanation.

15. What are implicit parameters? Name some instructions that have implicit parameters.

16. Why does the MIPS instruction set not contain a subtract immediate instruction?

17. Look at each MIPS pseudo-instruction and explain why it is a pseudo-instruction instead of a "real" instruction.

18. Give examples of how the MIPS instruction set exhibits each of the four design principles.

19. A colleague notices that the jump instruction `j label` can be replaced by a pseudo-instruction `beq R0, R0, label`. He then proposes eliminating the MIPS jump instruction *j* (the complier would then replace any `j` instructions with the corresponding `beq` instruction.)

    (a) Which design principles suggest that you should keep the `j` instruction? Why?

    (b) Which design principles suggest that you should eliminate the `j` instructions? Why?

20. Be able to discuss any of the design decisions we discussed in class in terms of the four design principles. This list includes, but is not limited to:

    - word size,
    - number of registers,
    - addressing modes (including when — or if — each is used),
    - instruction length (including the tradeoffs between fixed and variable width instructions), and
    - general purpose vs. special purpose registers.

21. Explain how the following design decisions are all inter-related:

    - word size
    - CISC vs. RISC
    - fixed vs. variable instruction size
    - load/store vs. register/memory
    - number and type of addressing modes
    - number of registers
    - size of registers

22. Why did x86 develop as a variable-width instruction set?

Name: _____

23. Consider a hypothetical computer with an instruction set of only two n-bit instructions. The first bit specifies the opcode, and the remaining $n-1$ bits specify one of the 2^{n-1} n-bit words of main memory. The two instructions are:

    - SUBS X: Subtract the contents of memory location X from the accumulator register, and store the result in both location X and the accumulator.
    - JUMP X: Unconditionally jump to location X.

    A word in main memory may contain either an instruction or a binary number in twos complement notation. Demonstrate that this instruction set is reasonably complete by showing how the following operations can be programmed using only the two operations:

    (a) Data transfer: Location X to accumulator, accumulator to location X

    (b) Addition: Add contents of location X to accumulator

    (c) Conditional branch

    (d) Logical OR