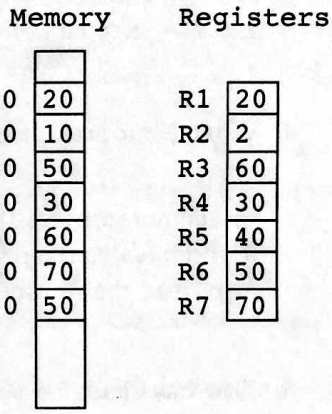
Practice Test 2 Solutions

February 29, 2016

Test 2 is Thursday, 2 March. No problems are due for credit; but, pay special attention to problems 1 and 13 from this practice test, as well as those problems on the performance homework that weren't assigned.

1. 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	52
ADD R1,R2, 40	immediate	42
ADD R1,R2, 40	memory direct	32
ADD R1, R2, R3	register indirect	72
ADD R1, R2, R3	register direct	62



2. List and explain the direct and indirect effects of a computer's word size (16-bit, 32-bit, 64-bit, etc.).

strongly suggests how big the registers should be. The register size, in turn, strongly suggests how big the virtual memory space should be. Also, the memory system is often optimized for the word size, thus, many (but not all) architectures design instructions to be multiples of the word size.

3. 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.

Immediate: `x = y + 6`
Register Direct: `x = y + z`, if `y` and `z` have already been loaded into registers.
Register Indirect: `x = y + *z` if `y` and `z` have already been loaded into registers.

Memory Direct: The first reference to a global variable may be memory direct, if the hardware supports such an addressing mode. Consider the code below:

```
.data

val1: 42
.text
lw $t1, val1
```

If the hardware supported a memory direct addressing mode, the instruction *may* be implemented like this: `lw, $t1, 0x10010000`. In reality, MIPS uses a displacement addressing mode because you can't put a 32-bit memory address in a 32-bit instruction. Intel places global data relative to the instruction pointer (`%rip`) to simplify the handling of shared libraries.

Memory indirect: The de-referencing of a global pointer may be memory indirect, if the hardware supports such an addressing mode. (See discussion above.)

4. What is an *accumulator*?

special-purpose register that serves as an implicit parameter to many arithmetic and logic instructions. It *accumulated* the results of a chain of calculations. (See p340 in the 7th edition of the Stallings text. Also, read the Accumulator page on Wikipedia.)

5. How has changing technology affected the way we design CPUs?

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

Implicit parameters are parameters that are not explicitly listed in the instruction. For example: In MIPS, the `jal` instruction stores the value of PC + 4 in register `$ra` (register 31). This register is specified implicitly by the instruction — it is not possible to explicitly specify the register in which PC + 4 is stored.

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

Because an `addi` with a negative immediate value will do the same thing.

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

9. 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?

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

- Regularity: Only 3 instruction formats. Instruction formats re-used where possible. All instructions 32-bits. All op-codes 6 bits
- Smaller is faster: Only 32 registers. Limited number of instructions. Limited number of addressing modes. Instructions limited to 32-bits.
- Common case fast: Limiting immediate values to 16-bits, and requiring a 2nd instruction for those cases that require more.
- Compromises: 2nd instruction format for immediate parameters. 3rd instruction format for jump instructions.

11. 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
- load/store vs. register/memory designs
- general purpose vs. special purpose registers.

12. 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

13. 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

14. The performance of a computer, in terms of how long it takes to run a program, can be expressed as:

$$\frac{seconds}{program} = \frac{instructions}{program} \times \frac{cycles}{instruction} \times \frac{seconds}{cycle}$$

(a) Show that the units in this equation cancel correctly.

(b) When discussing a single-cycle CPU, we used the formula $time = np$.

i. What term in the equation above corresponds to n? $\frac{instructions}{program}$

ii. What term in the equation above corresponds to p? $\frac{seconds}{cycle}$

iii. What happens to the remaining term ($\frac{cycles}{instruction}$)? In a single-cycle CPU, $\frac{cycles}{instruction}$ is always 1.