Name Answers

CE-1921-11 – Dr. Durant – Quiz 1
Spring 2016, Week 1

1. **Define** architecture
2. **List** the two locations that hold operands.
3. **Define** instruction.
4. **State** the bit width of every ARM instruction.
5. **State** the name of the software tool that translates high-level language to assembly language.
6. **List** two of the four modern architectural design rules.
7. **Define** source operands.
8. What is a **disadvantage** of a RISC architecture relative to a CISC architecture?
9. **Describe** the impact the number of instructions has on code density thus instruction storage size.
10. **Explain** what the following instruction does: SUB R3, R4, R5.

(1) Programmer's low-level view of computer
(2) registers + memory
(3) a basic operation that a processor is designed to do.
(4) 32
(5) compiler
(6) (a) regularity yields simplicity
(b) make the common fast
(c) small = fast
(d) design is choosing compromises
(7) The inputs to an operation
(8) Programs are longer; can't express more complex goals in single instructions
(9) ↑ instructions → ↑ bits to encode → ↓ dense code (per instruction) → ↑ storage size for 1 instruction
(10) ALU computes R4 - R5 + stores result in R3

Name Answers

CE-1921-12 – Dr. Durant – Quiz 1
Spring 2016, Week 1

1. **List** the two primary items that make up an architecture.
2. **Contrast** system architecture and micro-architecture.
3. **Define** instruction set.
4. **Describe** the difference between assembly language and machine language.
5. **State** the name of the software tool that translates assembly language to machine language.
6. **Define** mnemonic.
7. **Define** destination operand.
8. **Describe** the impact the number of instructions has on mnemonic encoding and thus circuit design.
9. **State** the number of ARM registers.
10. **Translate** the following Java/C statement into ARM assembly: a = b - c; Use R0, R1, and R2 for the variables, respectively.

(1) instruction + register set
(2) use of CPU chip in system vs. design of chip internals to meet architecture requirements
(3) all the instructions specified in an architecture
(4) assembly language uses mnemonic codes, that are easier for people to understand. These correspond to machine language instructions, the bit patterns recognized by hardware.
(5) assembler
(6) letters representing instruction (as opposed to bits), eg. "add"
(7) where the result of an instruction is written
(8) ↑ instructions → ↑ mnemonic length / complexity
→ ↑ hardware complexity
(9) 16
(10) SUB R0, R1, R2