

Exercise walk-through

6.1

Instruction set architecture

University Computer Science

Instruction set architecture: the stable contract

- 1 **Formats** R-type: opcode, rd, rs, rt, shift, funct - 28 bits used. I-type trades the third register for a 16-bit immediate.
- 2 **Range** Immediate spans -32 768 to 32 767, so 40 000 cannot encode; word offsets are multiples of 4, reaching 64 KiB from the base.
- 3 **Zero** addi R0, R0, 0 is the no-op; add R5, R0, R0 zeroes a register; beq against R0 needs no constant.

Instruction encodings

two formats, one 32-bit width

R-type



I-type



field widths in bits below each box - R-type: $4+4+4+4+4+8 = 28$, padded to 32; I-type: $4+4+4+4+16 = 32$

example: add R3, R1, R2 = 0001 0011 0001 0010 0000 00000001

Two formats, one width: R-type carries three registers and a function code; I-type trades the third register for a 16-bit immediate.

Exercises 2.1-2.3

2.1 Give the field values for `add R3, R1, R2` (opcode 1, funct 1): six numbers in format order. [3]

2.2 State the largest and smallest encodable `addi` immediate. [3]

2.3 Name the two instructions that touch data memory in a load-store ISA. [3]

Exercises 2.1-2.3

2.1 Give the field values for `add R3, R1, R2` (opcode 1, funct 1): six numbers in format order. [3]

2.1

- opcode 1, rd 3, rs 1, rt 2, shift 0, funct 1 ✓.

2.2 State the largest and smallest encodable `addi` immediate. [3]

2.2

- Largest 32 767; smallest -32 768 ✓.

2.3 Name the two instructions that touch data memory in a load-store ISA. [3]

2.3

- `lw` and `sw` ✓.

Exercise 3.1

3.1 The teaching ISA encodes `addi R5, R2, imm` with opcode 2.

Write the field values for `addi R5, R2, 60` in I-type order (including the unused pad), naming each field. Decode the I-type word whose fields read opcode 3, rt 6, rs 3, immediate 12 as a `lw` (mnemonic and operands). State which single field tells these two instructions apart, and why the R-type format also needs a funct field. [8]

Exercise 3.1

3.1 The teaching ISA encodes `addi R5, R2, imm` with opcode 2.

Write the field values for `addi R5, R2, 60` in I-type order (including the unused pad), naming each field. Decode the I-type word whose fields read opcode 3, rt 6, rs 3, immediate 12 as a `lw` (mnemonic and operands). State which single field tells these two instructions apart, and why the R-type format also needs a funct field. [8]

3.1

- `addi R5, R2, 60`: opcode 2, rt 5, rs 2, pad 0, immediate 60 = 111100_2 ✓.
- Opcode 3, rt 6, rs 3, imm 12 decodes to `lw R6, 12(R3)` ✓.
- The opcode alone tells them apart (2 versus 3) ✓.
- R-type reuses one opcode for many operations – the funct field completes the selection ✓.

Exercise 3.2

3.2 A program needs R7 to hold 40 000 before a loop, starting from $R0 = 0$.

State why one `addi` cannot do it. Write a two-instruction sequence that can (any correct pair, e.g. an `addi` of 20 000 then an `add` of the register to itself), giving each field's values. State the largest multiple of 4 reachable by one `lw` offset and what the ISA guarantees about R0 throughout.

[7]

Exercise 3.2

3.2 A program needs R7 to hold 40 000 before a loop, starting from $R0 = 0$.

State why one `addi` cannot do it. Write a two-instruction sequence that can (any correct pair, e.g. an `addi` of 20 000 then an `add` of the register to itself), giving each field's values. State the largest multiple of 4 reachable by one `lw` offset and what the ISA guarantees about R0 throughout. [7]

3.2

- 40 000 > 32 767: outside the 16-bit signed immediate ✓.
- E.g. `addi R7, R0, 20000` (opcode 2, rt 7, rs 0, imm 20000) then `add R7, R7, R7` (opcode 1, rd 7, rs 7, rt 7, shift 0, funct 1) ✓.
- Largest positive `lw` offset: 32 764 – the greatest multiple of 4 inside the signed 16-bit range; the field's full unsigned span covers $2^{16} = 64$ KiB of addresses around the base ✓.
- R0 is hardwired: every read yields zero and writes to it vanish ✓.