

# Solutions to HW 1

Jan 2002

## 1 1.1

Register r2 contains the value 28. The instruction `sw $r4, 8($r2)` stores the value in register r4 into the consecutive memory locations starting at (8 + value in r2), which is  $8+28 = 36$ . So, after the instruction is executed, the memory locations 36-39 will contain the value 44. No registers are altered in a store instruction.

## 2 1.2

In an R-type instruction, memory is not changed. This instruction adds the values present in registers r0 and r1 and stores the sum in register r2. So, register r2 will contain  $0 + 8 = 8$ .

## 3 1.3

The Add immediate instruction is similar to the add instruction, but an immediate value is added to a register value. So, the value in register r5 (36) is added to the immediate value (-4) and the result (32) is stored in register r6.

## 4 1.4

This instruction moves a 4-byte value word whose location is given by r3 (locations 16-19 contain 4) to the location given by  $r6 + 4 \times r1 = 52$ . Location 52 to 55 would contain the value 4 and r3 would contain the value 20.

## 5 1.5

An optimal assignment would be to have the following distribution:

2 bits for the Op-code.

8 bits for the function (R-type instructions) or immediate value (I-type)

6 bits for 2 register operands (3 + 3)

It takes 3 bits to specify the registers as there are 8 of them.

The assignment of Op-Codes could be as follows:

00 -> R-type instruction

01 -> Load

10 -> Store

11 -> BEQ

Note: The following method of assignment is sub-optimal:

200 R-format + 3 I format instruction = 203 Use 8 bits for Op-Code.  
Although we gain 2 more bits for R-format instructions, we lose 6 bits for I-format instructions, as previously, we used only 2 bits as opcode for I-format instructions. So, we could have only  $(16 - 8 - 2 \times 3) = 2$  bits for an immediate operand!

## 6 1.6

Given: 2 bytes for memory locations, 1 byte for opcode, 4 bytes for data  
LOCATIONS A, B AND C contain a, b and c.

Accumulator based machine:

```
LOAD B
ADD C
STORE A
ADD C
STORE B
LOAD A
SUB B
STORE D
```

Code Bytes =  $(1 + 2) \times 8 = 24$  bytes.

Data Bytes =  $4 \times 8 = 32$  bytes.

#### Memory-Memory

ADD A, B, C

ADD B, A, C

SUB D, A, B

Code Bytes:  $(1 + 3 \times 2) \times 3 = 21$  bytes

Data Bytes =  $3 \times 12 = 36$

#### Stack Machine

PUSH B

PUSH C

ADD

POP A

PUSH A

PUSH C

ADD

POP B

PUSH B

PUSH A

SUB

POP D

Code Bytes:  $(1+2) \times 9 + 1 \times 3 = 30$  bytes

Data Bytes:  $4 \times 9 = 36$  bytes

#### Load-Store Machine

LOAD r1, B

LOAD r2, C

ADD r3, r1, r2

STORE r3, A

ADD r1, r2, r3

```
STORE r1, B
SUB r4, r3, r1
STORE r4, D
```

Code Bytes:  $\text{ceiling}(1 + 0.5 + 2) \times 5 + \text{ceiling}(1 + 3 \times 0.5) \times 3 = 29$  bytes

Data Bytes =  $4 \times 5 = 20$

Thus, the total of (code + data) bytes is minimum for the load-store architecture based machine (49).