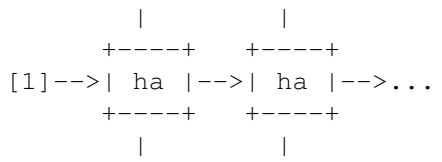
Computer Architecture: tutorial exercise answers

Exercise 2.1

(a) An array of halfadders with the carry-in of the least-significant element initialised to 1:



(b) Connect the array of halfadders to the output of the ALU in slide 4.10. Given that `s`, `d0` and `d1` are the control inputs for the ALU in slide 4.10 and `d2` denotes the carry-in of the array of halfadders, the control for the revised ALU becomes (`a` and `b` are the two multi-bit inputs to the ALU, as shown in slide 4.10):

```
---------------------------------
control inputs
  s d0 d1 d2    operation selected
---------------------------------
  1  1  0  1    a + b + 1
  1  1  1  1    a - b + 1
  1  1  0  0    a + b
  1  1  1  0    a - b
---------------------------------
```

Exercise 2.2

The key is to distinguish between the *instruction operand* which is part of the instruction, and the *data operand* which is 4 bytes long. For instance for the `load b` instruction, there are 3 bytes for the instruction (1 byte for opcode and 2 bytes for a memory address), and 4 data bytes (for the data operand).

- Accumulator

Instruction		code bytes	data bytes
load	b	3	4
add	c	3	4
store	a	3	4
add	c	3	4
store	b	3	4
neg		1	0
add	a	3	4
store	d	3	4
Total		22	28

Code size is 22 bytes, and memory bandwidth is 22 + 28 = 50 bytes.

- Stack

Instruction	code bytes	data bytes
push b	3	4
push c	3	4
add	1	0
dup	1	0
pop a	3	4
push c	3	4
add	1	0
dup	1	0
pop b	3	4
neg	1	0
push a	3	4
add	1	0
pop d	3	4
Total	27	28

  Code size is 27 bytes and memory bandwidth is 27 + 28 = 55 bytes.

- Memory-Memory

Instruction	code bytes	data bytes
add a,b,c	7	12
add b,a,c	7	12
sub d,a,b	7	12
Total	21	36

  Code size is 21 bytes and memory bandwidth is 21 + 36 = 57 bytes.

- Load/Store

Instruction	code bytes	data bytes
load $1, b	4	4
load $2, c	4	4
add $3, $1, $2	3	0
store $3, a	4	4
add $1, $2, $3	3	0
store $1, b	4	4
sub $4, $3, $1	3	0
store $4, d	4	4
Total	29	20

  Code size is 29 bytes and memory bandwidth is 29 + 20 = 49 bytes.

The Load/Store machine has the lowest amount of data traffic. It has enough registers that it only needs to read and write each memory location once. On the other hand, since all ALU operations must be separate from the loads and stores, and all operations must specify three registers or one register and one address, the Load/Store machine has the largest code size. The Memory-Memory machine, on the other hand, is at the other extreme. It has the fewest instructions (though also the largest number of bytes per instruction), and the largest number of data accesses.