

15-740/18-740 Computer Architecture

Homework 3

Due Friday, September 30, at 12:00 PM

1. Semantic Gap

In class, we talked about the ISA and semantic gap. One potential option is to place the ISA closer to the high level language (with a correspondingly small semantic gap) and translate the program into something that has a larger semantic gap (i.e., closer to the control signals). For example, Transmeta used the x86 ISA as the virtual ISA but translated the x86 binary to an internal VLIW ISA which formed the real hardware/software interface. This translation layer could be built in hardware and software, what are the advantages and disadvantages of each?

2. Address Mapping

In class, we learned that though the address space of a program may seem contiguous to the programmer, in the underlying hardware, data are mapped to banks in main memory. In this problem, we will examine how different data mapping schemes may affect performance.

Let's say you want to run the following code on a vector processor with $VLEN = 2$, and a word size of a byte:

```
VLD v0 ← 0x0
VLD v1 ← 0x2
```

- (a) Consider a system with only one bank, whose address is mapped in the following way:

Bank 0
Byte 0x0
Byte 0x1
Byte 0x2
Byte 0x3

Assuming the processor can request **and** receive one byte of data from a single bank each cycle (the bus connecting the banks and the processor is shared among all the banks), and a bank can return one byte of data every four cycles, how many cycles would it take to execute the previously-mentioned program?

In other words, each cycle, the processor can issue one load for one byte to a bank that is not processing a prior load. The same cycle the processor issues the load to the bank, the bank can start processing the load.

- (b) Now, consider a system with two banks, whose address is mapped in the following way:

Bank 0	Bank 1
Byte 0x0	Byte 0x1
Byte 0x2	Byte 0x3

Under the same assumptions as before, how many cycles would it take to execute the program?

(c) Alternatively, with two banks, the data could be mapped in the following way:

Bank 0	Bank 1
Byte 0x0	Byte 0x2
Byte 0x1	Byte 0x3

Under this mapping, how many cycles would it take to execute the program?

(d) Finally, a more costly, but potentially higher-performance, option is to map each byte of data to its own bank:

Bank 0	Bank 1	Bank 2	Bank 3
Byte 0x0	Byte 0x1	Byte 0x2	Byte 0x3

How many cycles would such a system take to execute the program?

3. Vector Processing

Next, consider a similar program and the same assumptions as above:

```
VLD  v0 ← 0x0
VLD  v1 ← 0x2
VADD v3 ← v0, v1
```

Assume bytes of data can be forwarded from the vector loader to the vector adder as soon as they are received from main memory **and the adder takes two cycles to complete**. How many cycles would this program take to execute under each of the four mappings presented earlier if:

- (a) The vector add **can not** be pipelined?
- (b) The vector add **can** be pipelined?

4. CISC versus RISC

We learned about a relatively complex x86 instruction in class, *REP MOVS*, which moves a string from one location to another. The VAX instruction set architecture included some interesting (and complex) instructions such as *CRC* to calculate a cyclic redundancy check, *INSQUEUE* and *REMQUEUE* to insert into and remove from doubly-linked list, *INDEX* to perform array indexing and bounds checking, and *LOCC* to locate a given ASCII character in a null-terminated ASCII character string starting at a particular address (i.e., *LOCC rI, rC, rS* would store in *rI* the first index of the character stored in *rC* in the string starting at the address stored in *rS*, or, if the character is not found, the index of the null character at the end of the string).

The ARM instruction set architecture¹ contains relatively simple instructions in comparison to x86, but that does not limit its functionality. Let's say you've been hired by the startup Spansneta to design a software translation layer from VAX to ARM for use with their next chip. Using the referenced ARM ISA quick reference card, translate the VAX instruction *LOCC r1, r2, r3* (mentioned above) to the ARM ISA.

¹http://infocenter.arm.com/help/topic/com.arm.doc.qrc0001m/QRC0001_UAL.pdf