Lecture C6: Computer Architecture

Response to 'Muddiest Part of the Lecture Cards'

(46 respondents, out of 64students)

1) ***"Help, I am confused about everything"?*** (and similar comments) (4 students)
Visit the Monday 4-5pm office hour session in the TA office. Another alternative is to talk to the grad TAs Howie and Carl, they have mentioned the possibility of starting some kind of tutoring sessions.

2) ***What about this info should we memorize? Why are we learning this info?*** (2 students)
All material in the reading instructions should be read. I only cover a subset of that material in class / lecture notes. The material I cover in class is the information I find the most important.

Most aerospace systems involve writing software at both the embedded level as well as the application level. The topics covered in lectures are meant to give you a first look at these topics.

3) ***How does computer architecture affect our programming in Ada?*** (2 students)
The architecture of the computer affects the size of the object code generated when you compile an Ada program. When dealing with embedded mission critical systems, the size of your code (assuming that design and implementation were correct in the first place), will determine the time your program takes to execute. It then becomes critical for you to understand the underlying architecture to be able to build a predictable system.

4) ***By what computer manufacturer was the PowerPC developed?*** (1 student)
As far as I know, the PowerPC was created by *AIM* (the 1991 Apple-IBM-Motorola alliance). The PowerPC was the CPU part of the *AIM platform*, and is the only part o that platform that still exists today.

5) ***What is difference between a bit and a byte?*** (1 student)
A bit is an information unit. A single bit is a 0 or a 1, or a true or a false, or an on or an off, or any other mutually exclusive states. A byte is a collection of bits, usually 8 bits.

7) ***How does the computer go from 1s and 0s to actual commands?*** (1 student)
The instruction set is actually coded in the forms of 0's and 1's. The coding scheme used to take in a bit pattern and extract the instruction is also defined by the architecture of the machine. For example, the instructions for the machine architecture seen in class uses a pattern that is 16 bits long, the first four bits specify the opcode (operation to be performed) and the remaining 12 bits define the operands (the elements on which the operation is performed). Most programmers only see the mnemonics (assembly language). Hardly anybody codes in 0's and 1's because the instructions sets are complex and it is very

easy for the programmer to make a mistake and extremely hard to find the mistake. In the next lecture we will address mnemonics in greater detail.

8) ***What are advantages and disadvantages of a multiprocessor machine?*** (1 student)
The primary advantage of having a multi-processor machine is more power. On the other hand, to exploit this additional power, the usage has to be managed, increasing the complexity of the operating system and/or programming language.

9) ***RISC vs CISC?*** (2 students)
RISC: Reduced Instruction Set Computer: RISC computers are build under the premise that it is possible to build more complex operations from simpler ones. This simplification forces the programmer to implement the more complex functions himself or herself.

CISC: Complex Instruction Set Computer: provide more complex operations than RISC computers. This does not mean the programmer is provided with everything, just more than RISC.

10) ***PC vs IR?*** (1 student)
PC = Program Counter = Special Purpose register within the CPU that contains the address of next instruction to be executed

IR = Instruction Register = Special Purpose register that contains the opcode of the instruction currently being executed within the CPU

11) ***von Neumann bottleneck?*** (1 student)
von Neumann computers spend a lot of time moving data to and from the memory. This slows the computer down. One way to get around this is to introduce more busses in the system, for example one bus for instructions, one bus for data and so on.

12) ***I do not understand Assembly language?*** (1 student)
That is the topic of tomorrow's lecture. Also see the answer for 7 above.

13) ***How do we know that address cells 00-FF account for 256 cells?*** (1 student)
00 and FF are hexadecimal numbers. FF in base 16 equals 255 in base 10 (decimal number system), thus 00-FF in hex = 00-255 which gives us 256 memory cells.

14) ***Why do I never hear about the Registers in the CPU when I hear about computers and their performance?*** (1 student)

Most people associate the clock speed of the computer with it performance. But from today's lecture, i hope that you realized that the true bottleneck is in the bus speed. That should give you a sense of the performance capability of the computer. Increasing the number of registers will not necessarily increase the performance of your computer. The issues regarding performance are more tightly coupled with the cache as well.

15) ***Control signal specification, micro program?*** (1 student)

The CPU is actually a chip, with pins that connect it to the buses and power. The control program spec/microprogram control the physical chip activities such as reading from a pin or writing to a pin (changing voltage levels).

16) ***Where is the cache located?*** (1 student)
The cache is located both on the processor as well as on the bus. The cache on the CPU however is really expensive.

17) ***Does information need to be in a register before it can be used? Can't it be used directly from memory? What exactly does registers do?*** (1 student)

Registers are named memory locations on the CPU. All the operations that can be performed the CPU are performed using registers. Hence data has to be read into the register before it can be manipulated by the CPU.

18) ***Can you review Device Controllers again, and the von Neumann architecture?*** (1 student)
I will start tomorrow's lecture with a short 'recap' of today's lecture.

19) ***What makes it difficult for the ALU to perform division?*** (1 student)
If you sit down and think about the algorithm behind a division you will find that it is a bit trickier than just performing addition or subtraction of integers. An idea behind building hardware that can perform division could be to repeatedly subtract the divisor and shift as appropriate. Dividend = Divisor*Quotient +Remainder. Which gives us that we are going to need hardware to 1) hold divisor and shift it right, 2) hold the remainder, 3) hold quotient and shift it left, 4) subtract the divisor from the current result, 5) control the whole process.

20) ***How is information relayed within the CPU (bus in there too?)?*** (1 student)
Yes, there are busses within the CPU.

21) ***With Cache memory and other short term memory, how does the computer decide what to store there and when to erase it?*** (1 student)

Managing memory is the operating systems responsibility. The programmer can (sometimes) specify if certain variables/values need to be stored in the cache for the duration of he program.

22) ***The way you have the bus drawn shows that Video Controllers have DMA, can the Device Controllers interact with each other?*** (1 student)

Some devices can communicate using Direct Memory Access (DMA). This allows the CPU cycles to be used effectively for computations rather than data transfer.

23) ***For pipelined CPU: It seems like it faster than von Neumann architecture, but wouldn't flow/ speed be "held up" if one of the units (fetch, decode, or execute) is running too slow?*** (1 student)
That is a correct observation!

24) ***"No mud"*** (21 students)
Good :o)