COMPSYS304 Notes 2017

Theodore Oswandi

July 26, 2017

1 Lecture 1

1.1 Improvements

- Semiconductor technology and computer architecture has increased lots in last 50 years
- Performance, which has also increased can be measured from standardised benchmarks
- Clock rate/frequency has also increased considerably in this time.

1.2 Computer Architecture

ISA Boundary between hardware and software

Oragnisation high level computer design aspects

Hardware detailed logic and circuit design

Note: You want to separate your instruction set from implementation

1.3 Memory Organisation

- See memory as single 1D array
- Address is index of this array, points to byte of memory.
- Memory Access Time: time to read data to/from memory

- Memory Speed != Processor speed.
- Fast memory is very expensive. Heirarchy used to maintain fluid functionality and keep things cheap.

Processor Registers

- Smallest and fastest memory for CPU
- about 32-64 of them.
- Each are 32/64bits in size.
- Nanosecond access time

Cache Memory

- Slower than register
- 8-256k
- Few nanoseconds access time
- Levels to this as well. L1, L2, L3 cache used in multiprocessor systems.

Main Memory

- Slower than cache
- Megabytes to gigabytes of size.
- Tens of nanoseconds lookup time.

1.4 Instruction Set Architecture (ISA)

ISA is interface between hardware and low level software.

Modern ISA include 80x86, MIPS, ARM

1.4.1 Using Fixed ISAs

Uses old instruction set (1970s), also used with extensions to enable newer technologies such as internet, etc...

Advantages

- AMD/Intel both have same ISA but different implementation.

Disadvantages

- power consumption is higher than things like iPad which use different ISA and consume a lot less power
- Also prevent some new innovation since it is so widely used in today's world.

1.4.2 ISA Design

Need to ask:

- What operations do the CPU need to do?
- How to provide data for given operations?
- How to store results of these calculations?

Need to define:

- Instruction Format and Encoding
- Data types and their sizes
- Location of operands and where to store results

Operands and Opcodes To carry out these calculations, an **opcode** must be defined to define these calculations. Upon these opcodes, zero to three **operands** are used for data inputs and result outputs.

1.5 Architecture Types

1.5.1 Stack Base Architecture

- Top of stack will contain result of operation.
- If receive ADD then processor knows next 2 inputs contain 2 numbers that need to be added.
- PUSH add something to top of stack.
- POP use value in top of stack.
- JVM designed to use Stack based architecture.
- ADD function has no operators. Operates on last 2 loaded values.

1.5.2 Accumulator Based Archictecture.

- Using inputs from memory.
- Not used anymore today. Used in 1970s
- ADD function takes one operator, $mem_address$

1.5.3 Register Memory Architecture

- Currently used today as x86
- Uses register for input as well as access values from memory.

1.5.4 Register-Register Architecture

- Operands from register.
- LOAD and STORE from memory too.
- Need to specify destination register for output.

A(1000) + B(2000) = C(3000) Stack Based Architecture PUSH 1000 PUSH 2000 ADD POP 3000

Accumulator Based LOAD 1000 ADD 2000 STORE 3000

Register Memory LOAD R2, 1000 ADD R1, R2, 2000 STORE R1, 3000

Register Register LOAD R2, 1000 LOAD R3, 2000 ADD R1, R2, R3 STORE R1, 3000

ISA CLASSES RISC Reduced

CISC Complex

EPIC