INSTRUCTION SET ARCHITECTURE

Mahdi Nazm Bojnordi

Assistant Professor

School of Computing

University of Utah

Overview

- Homework 1 due on Jan 17th (midnight)
- Homework 2 will be released tonight

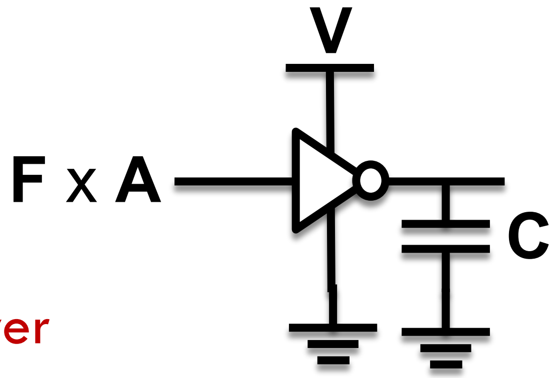
- This lecture
  - Recap CPU power and energy
  - Instruction set architecture (ISA)

Recall: CPU Power and Energy

- All consumed energy is converted to heat
  - CPU power is the rate of heat generation
  - Excessive peak power may result in burning the chip
- Static and dynamic energy components
  - $\text{Energy} = (\text{Power}_{\text{Static}} + \text{Power}_{\text{Dynamic}}) \times \text{Time}$
  - $\text{Power}_{\text{Static}} = \text{Voltage} \times \text{Current}_{\text{Static}}$
  - $\text{Power}_{\text{Dynamic}} = \text{Capacitance} \times \text{Voltage}^2 \times (\text{Activity} \times \text{Frequency})$

Power Reduction Techniques

- Reducing capacitance (C)
  - Requires changes to physical layout and technology
- Reducing voltage (V)
  - Negative effect on frequency
  - Opportunistically power gating (wakeup time)
  - Dynamic voltage and frequency scaling
- Reducing frequency (F)
  - Negative effect on CPU time
  - Clock gating in unused resources
- Points to note
  - Utilization directly effects dynamic power
  - Lowering power does NOT mean lowering energy

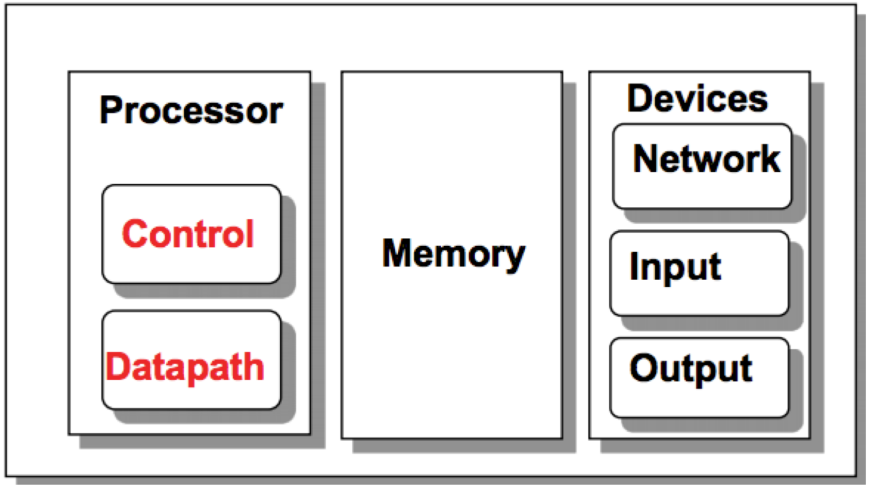


Instruction Set Architecture

- The key to program/use a microprocessor
  - The language of the hardware defines the hardware/software interface
  - Stored-program concept (von Neumann)
  - What are the principles for ISA design

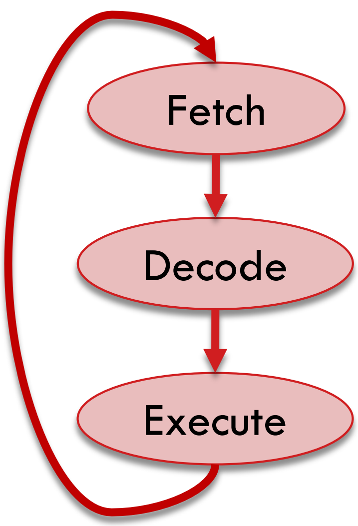
Instruction Set Architecture

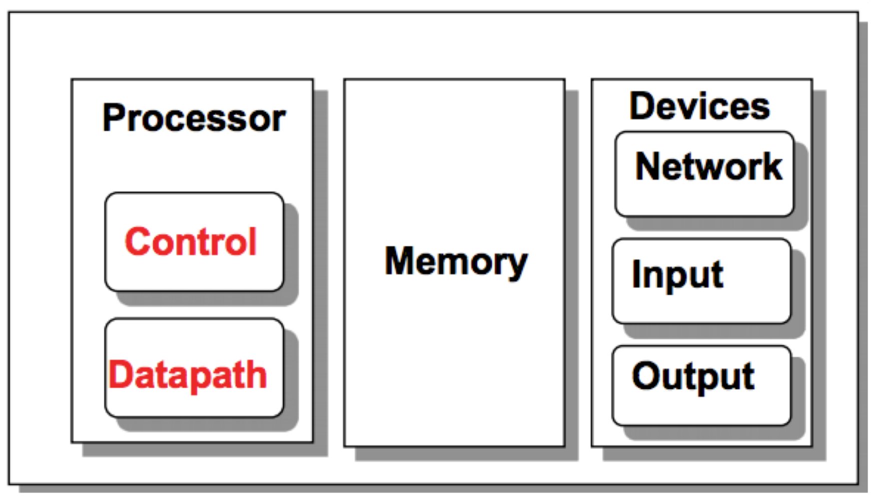
- A program (in say, C) is compiled into an executable that is composed of machine instructions
- Java programs are converted into portable bytecode that is converted into machine instructions during execution (just-in-time compilation)

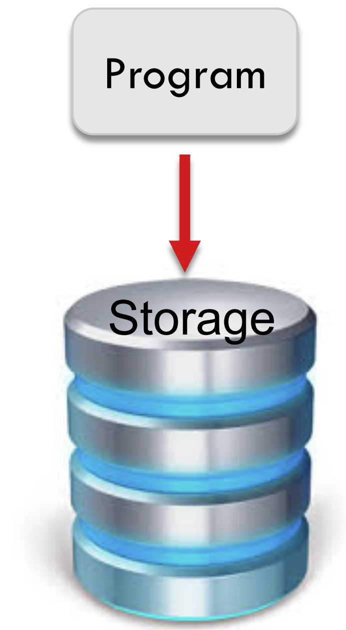


Instruction Set Architecture

- A program (in say, C) is compiled into an executable that is composed of machine instructions
- Java programs are converted into portable bytecode that is converted into machine instructions during execution (just-in-time compilation)







Quick Review: Data Representation

- Smallest unit of representing information in conventional computers is bit
  - Only two states: 0 and 1
- Multibit representation units are used to increase the number of states
  - Every group of 8 bits is called a byte representing 256 states
  - Multiple bytes form a word
    - 4-byte word or
    - 8-byte word in more modern processors

Instruction Set Architecture

- keep the hardware simple – the chip must only implement basic primitives and run fast
- keep the instructions regular – simplifies the decoding/scheduling of instructions
- MIPS instruction set architecture
  - Other examples are ARM, x86, IBM power, etc.
- Complex vs. simple instructions
  - Which one is better?

Example MIPS Instruction

- C code
  - High level language

```
a = b + c;
```

- Assembly code
  - Human friendly machine instruction

```
add   a, b, c      #  a is the sum of b and c
```

- Machine code
  - Hardware friendly machine instruction

```
00000010001100100100000000100000
```

Example MIPS Instruction

- Translate the following C code to assembly

```
a = b + c + d + e;
```

Example MIPS Instruction

- Translate the following C code to assembly

a = b + c + d + e;

- Assembly

```
add  a, b, c
add  a, a, d
add  a, a, e
```

```
add  a, b, c
add  f, d, e
add  a, a, f
```

Example MIPS Instruction

- Translate the following C code to assembly

a = b + c + d + e;

- Assembly

```
add  a, b, c
add  a, a, d
add  a, a, e
```

```
add  a, b, c
add  f, d, e
add  a, a, f
```

- Translate this one

f = (g + h) – (i + j);

Example MIPS Instruction

- Translate this one

$$f = (g + h) - (i + j);$$

- Assembly

```
add  f, g, h
sub  f, f, i
sub  f, f, j
```

```
add  t0, g, h
add  t1, i, j
sub  f, t0, t1
```

- In summary
  - operations are not necessarily associative and commutative
  - More instructions than C statements
  - Usually fixed number of operands per instruction