INSTRUCTION SET ARCHITECTURE

Mahdi Nazm Bojnordi

Assistant Professor

School of Computing

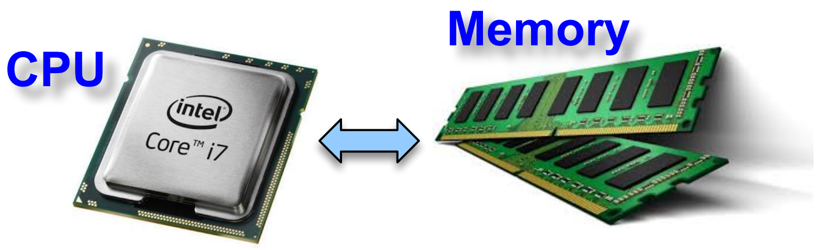
University of Utah

Computer Organization

- Classic components of a computing system

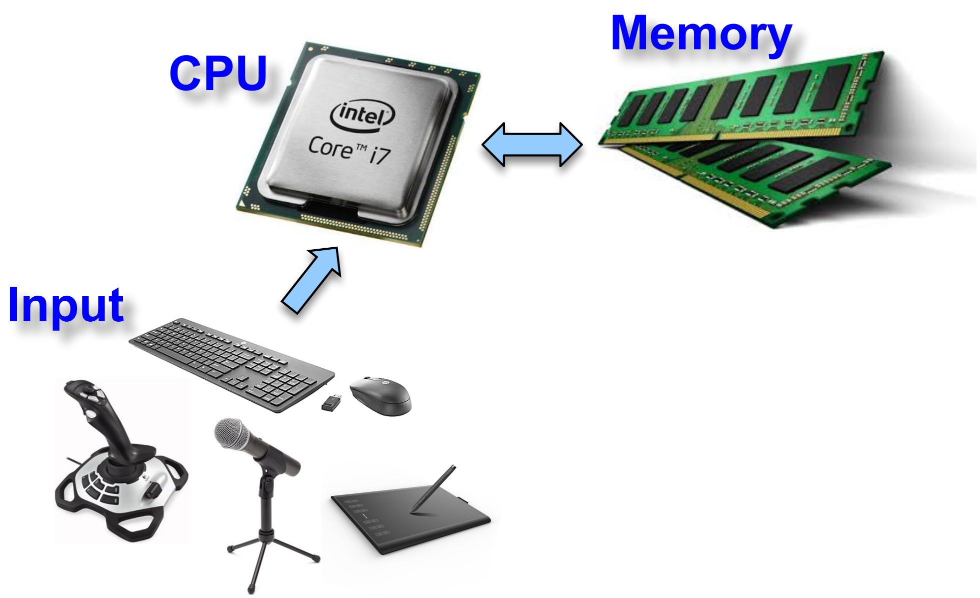
Computer Organization

- Classic components of a computing system



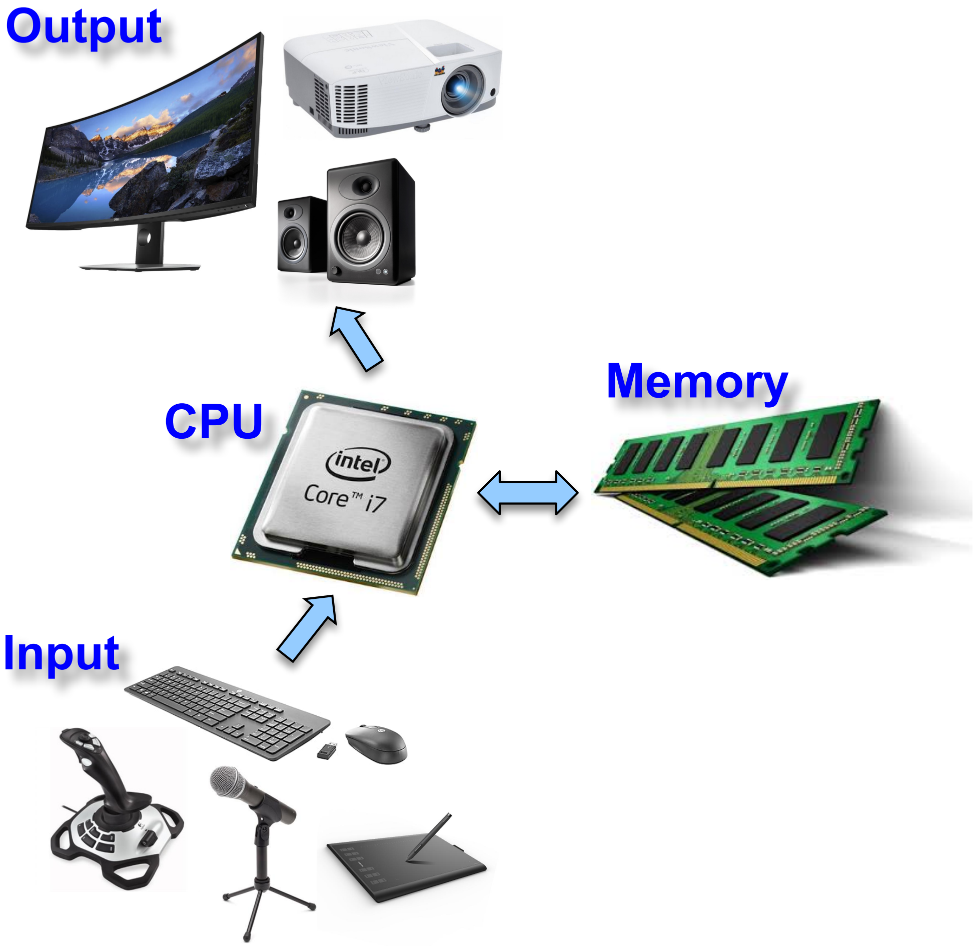
Computer Organization

- Classic components of a computing system



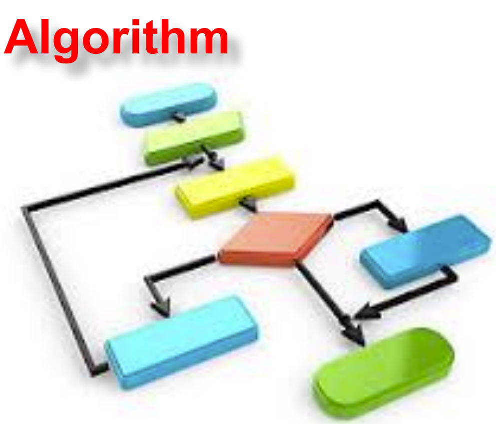
Computer Organization

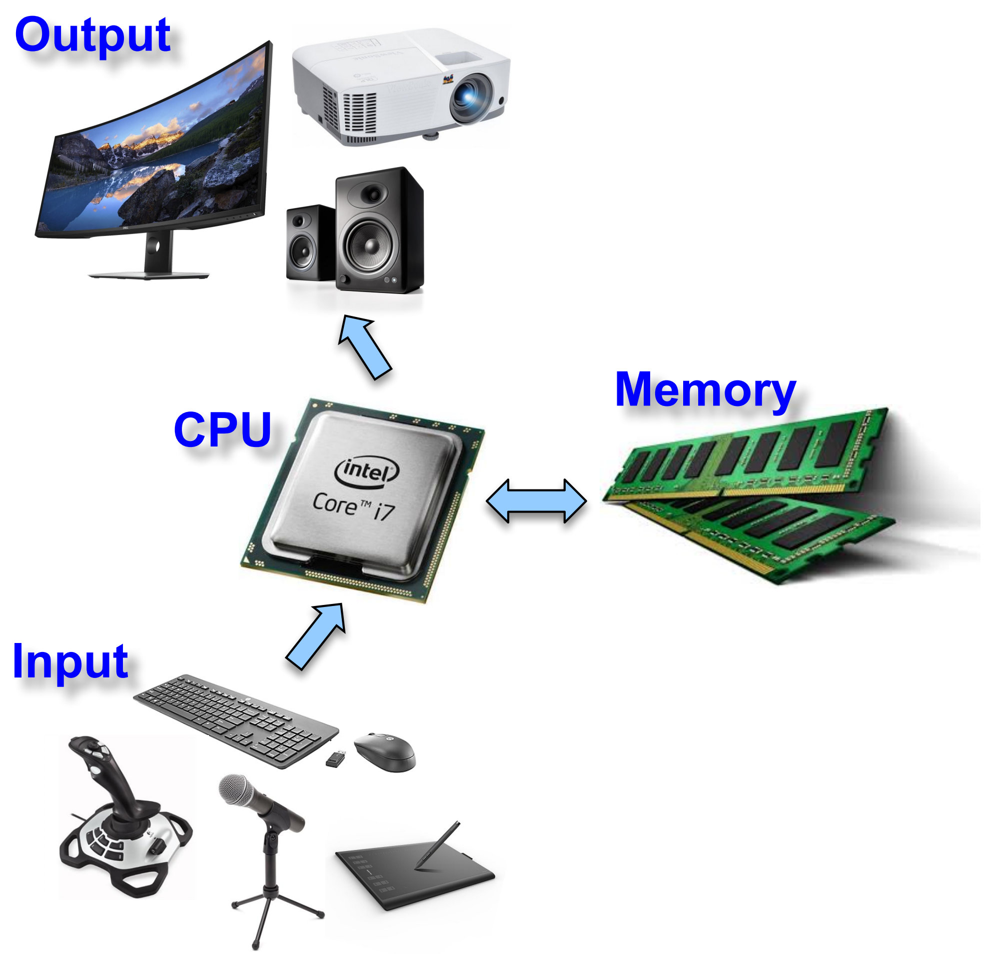
- Classic components of a computing system



Computer Organization

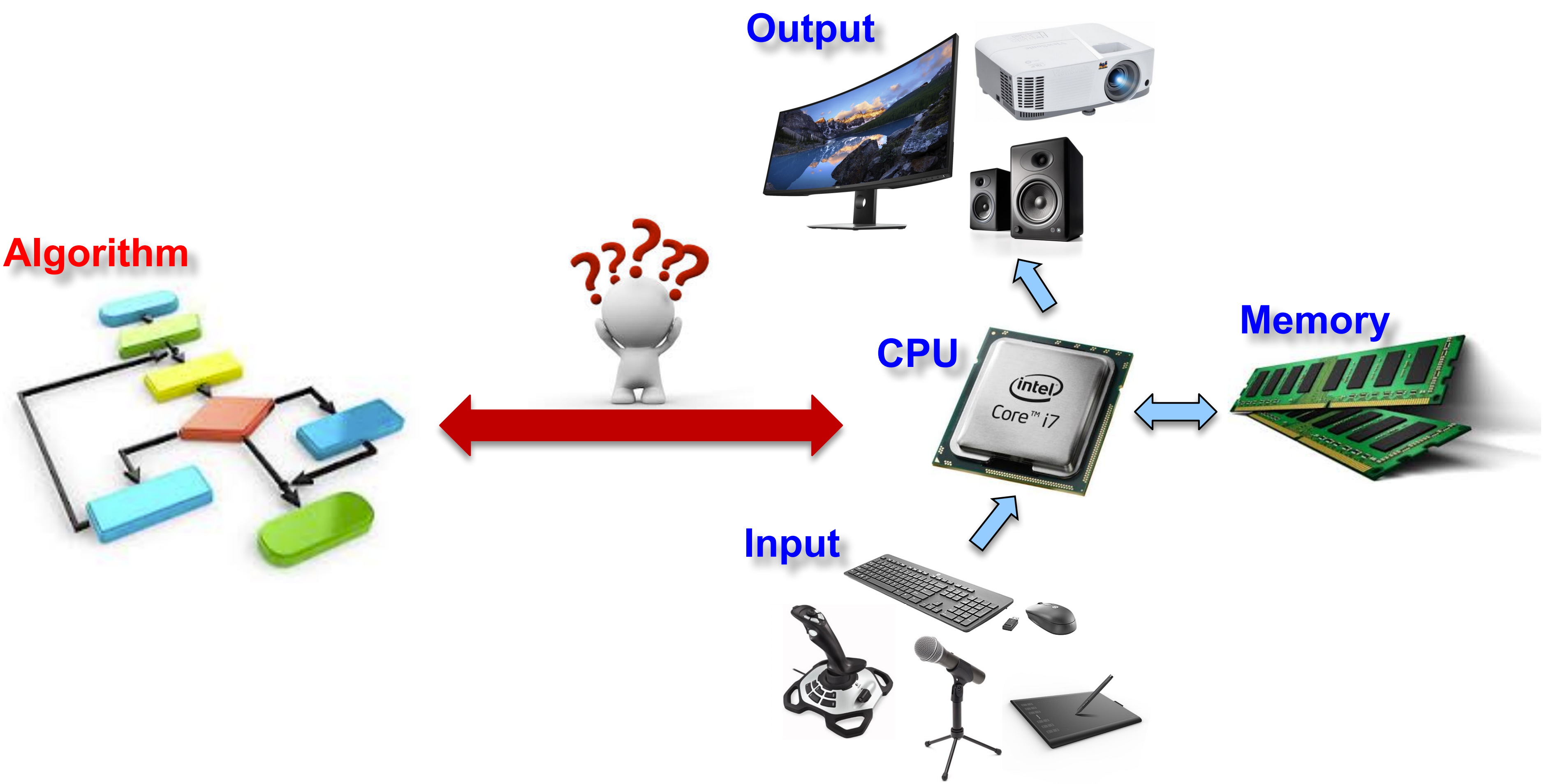
- Classic components of a computing system





Computer Organization

- Classic components of a computing system



Instruction Set Architecture

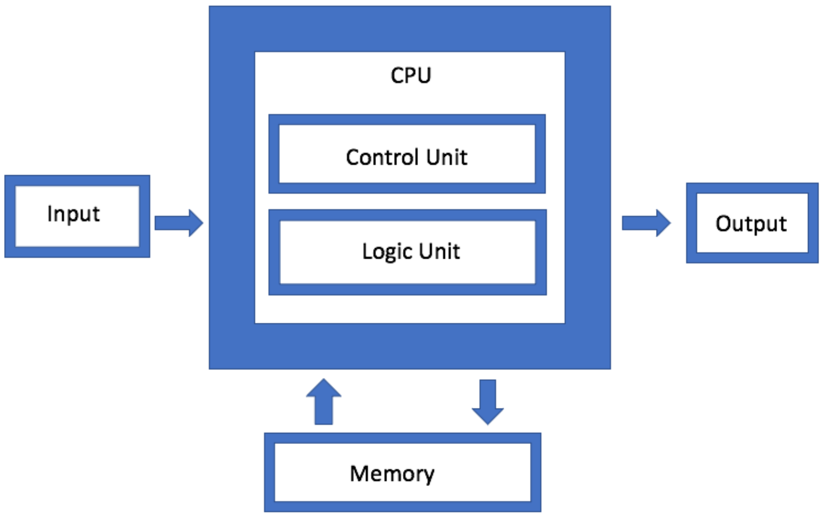
- The key to program/use a microprocessor
  - The language of the hardware defines the hardware/software interface
  - ISA is a contract between software and hardware

Software

Hardware

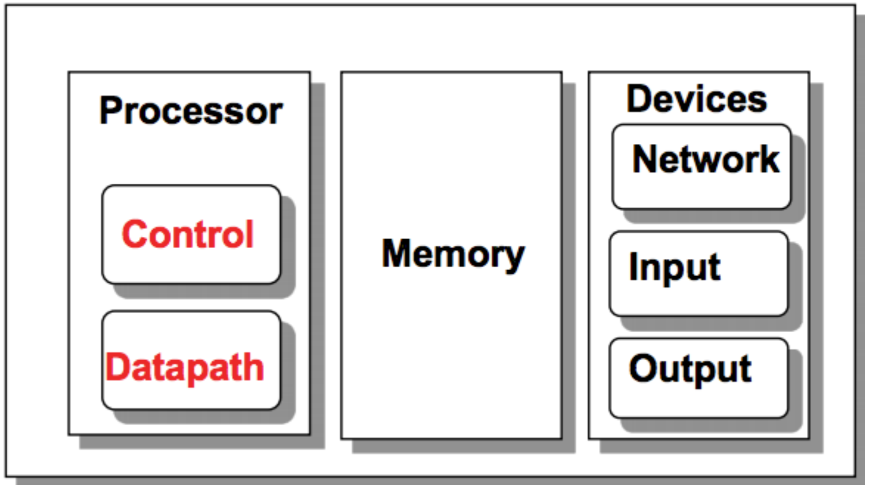
Instruction Set Architecture

- The key to program/use a microprocessor
  - The language of the hardware defines the hardware/software interface
  - ISA is a contract between software and hardware
  - Stored-program concept (von Neumann)



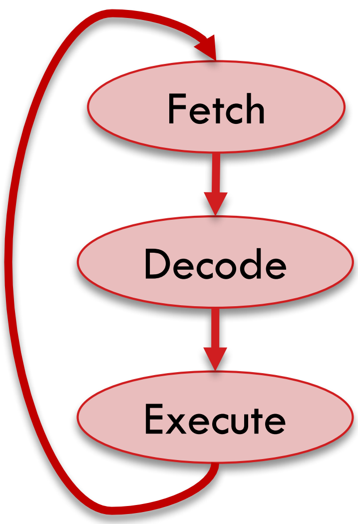
Instruction Set Architecture

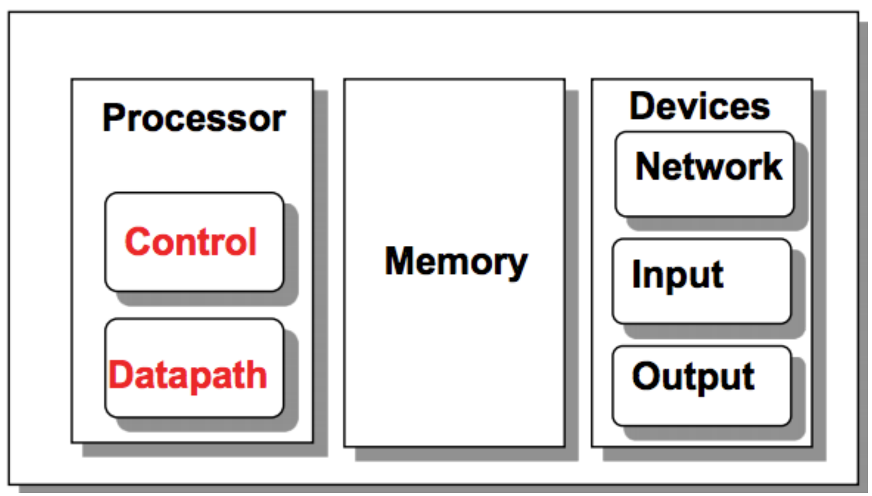
- A program (in say, C) is compiled into an executable that is composed of machine instructions
- Java programs are converted into portable bytecode that is converted into machine instructions during execution (just-in-time compilation)

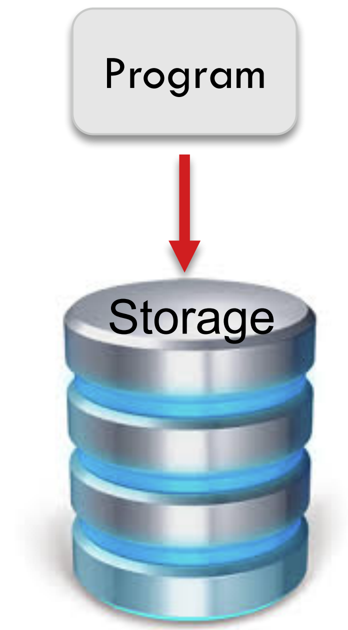


Instruction Set Architecture

- 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)







Data Representation

- Smallest unit of representing information in conventional computers is bit
  - Only two states: 0 and 1

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

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

Data Conversion

- Decimal is the most human-friendly base for presenting numbers
  - Example: 8163
- Convert decimal to binary (machine-friendly)
  - Through a series of divisions
  - Example: 1111111100011

Data Conversion

- Decimal is the most human-friendly base for presenting numbers
  - Example: 8163
- Convert decimal to binary (machine-friendly)
  - Through a series of divisions
  - Example: 1111111100011

Find the binary representation of 8163 through a series of divisions by 2.

Quotient	4081	2040	1020	510	255	127	63	31	15	7	3	1	0
Remainder	1	1	0	0	0	1	1	1	1	1	1	1	1

Answer: 1111111100011_{bin}

Data Conversion

- Decimal to Hexadecimal
  - Example: 8163

Find the hexadecimal representation of 8163 through a series of divisions by 16.

Quotient	Remainder
510	3
31	14
1	15
0	1

Value	Hex Digit
0	0
..	...
9	9
10	A
11	B
12	C
13	D
14	E
15	F

Answer: $1FE3_{hex}$

Data Conversion

- Decimal to Octal
  - Example: 8163

Find the hexadecimal representation of 8163 through a series of divisions by 8.

Quotient	Remainder
1020	3
127	4
15	7
1	7
0	1

Answer: 17743_{oct}

Conversion To Decimal

- From Binary (1111111100011)
  - $1x2^0 + 1x2^1 + 0x2^2 + 0x2^3 + 0x2^4 + 1x2^5 + 1x2^6 + 1x2^7 + 1x2^8 + 1x2^9 + 1x2^{10} + 1x2^{11} + 1x2^{12} = 8163$

Conversion To Decimal

- From Binary (1111111100011)
  - $1x2^0 + 1x2^1 + 0x2^2 + 0x2^3 + 0x2^4 + 1x2^5 + 1x2^6 + 1x2^7 + 1x2^8 + 1x2^9 + 1x2^{10} + 1x2^{11} + 1x2^{12} = 8163$
- From Hexadecimal (1FE3)
  - $3x16^0 + Ex16^1 + Fx16^2 + 1x16^3 = 3x16^0 + 14x16^1 + 15x16^2 + 1x16^3 = 8163$

Conversion To Decimal

- From Binary (1111111100011)
  - $1x2^0 + 1x2^1 + 0x2^2 + 0x2^3 + 0x2^4 + 1x2^5 + 1x2^6 + 1x2^7 + 1x2^8 + 1x2^9 + 1x2^{10} + 1x2^{11} + 1x2^{12} = 8163$
- From Hexadecimal (1FE3)
  - $3x16^0 + Ex16^1 + Fx16^2 + 1x16^3 = 3x16^0 + 14x16^1 + 15x16^2 + 1x16^3 = 8163$
- From Octal (17743)
  - $3x8^0 + 4x8^1 + 7x8^2 + 7x8^3 + 1x8^4 = 8163$

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