Data Manipulation

10/7/2009

Che-Rung Lee

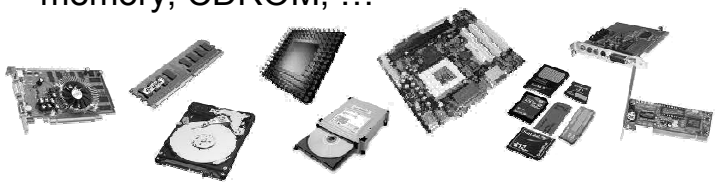
What is a computer?

- Monitor, case, keyboard, mouse, speaker, scanner, webcam, printer, …

What's inside?

Inside the case

- CPU, motherboard, adaptors, hard disk, memory, CDROM, …



We are going to talk about those.

Central Processing Unit (CPU)

- An electronic circuit that can execute computer programs
  - Intel i7
  - AMD K10
  - IBM Cell
  - ARM Acorn
  - Sun SPARC

- To understand CPU, we need to know what computer programs are.

Outline

- Store program concept
- Machine language
- Program execution
- Peripheral devices
- Parallel architectures

Stored Program Concept

"*The final major step in the development of the general purpose electronic computer was the idea of a stored program…*"
Brian Randell

What're the differences?

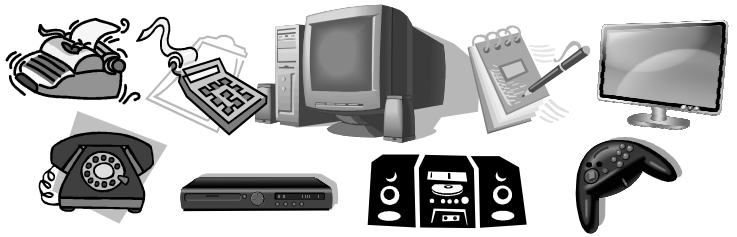
TV: you can watch different channels.

麵包機: you can make different food.

Swiss knife: you can use different tools

Computer: you can …

Magic box



- You can add more functions to it. How?
  - Program is like data to be input to computers.
- It can perform multiple functions at a time
  - We will talk about this in the OS lesson.

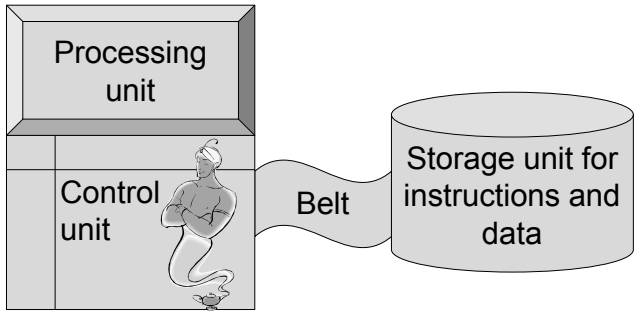
Store program concept

- Program: a sequence of instructions
- Store program concept: a program can be encoded as bit patterns and stored in main memory. From there, the CPU can extract the instructions and execute them.
- Advantage: programmable
  - We can use a single machine to perform different functions.

Problems

- How to convert instructions to operations?
  - This is like Harry Porter's spell.
- There should be a control unit.
  - To control which function to perform.
  - To control which data to be operated.
  - How can the control unit understand the instructions?
- What function units should be included?
  - CD players, game console, calculators, …?

Outline of the magic box



von Neumann architecture

general purpose electronic computer

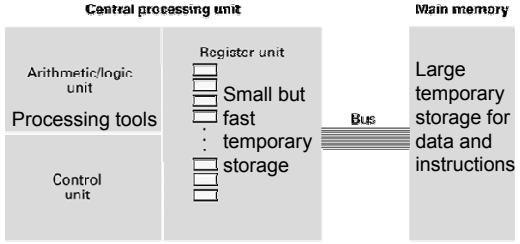


Figure 2.1 **CPU and main memory connected via a bus**

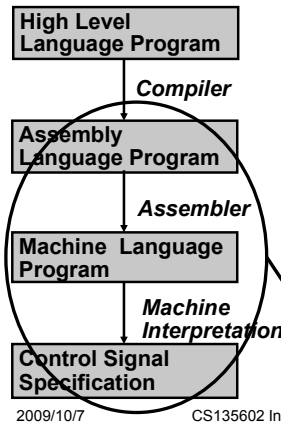
Machine Language

What to do

+

Specified information

Computer programs




```
temp = v[k];
v[k] = v[k+1];
v[k+1] = temp;
```

You are learning it in CS1355

```
lw    $15,  0($2)
lw    $16,  4($2)
sw    $16,  0($2)
sw    $15,  4($2)
```

You will learn it in CS2410

```
0000 1001 1100 0110 1010 1111 0101 1000
1010 1111 0101 1000 0000 1001 1100 0110
1100 0110 1010 1111 0101 1000 0000 1001
0101 1000 0000 1001 1100 0110 1010 1111
```

This will be taught in CS4100

We are going to talk about those.

ALUOP[0:3] <= InstReg[9:11] & MASK

Example: a = b + c

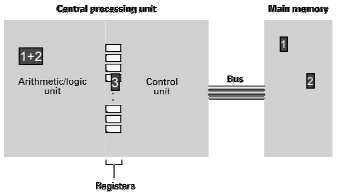
Step 1. Get one of the values to be added from memory and place it in a register.

Step 2. Get the other value to be added from memory and place it in another register.

Step 3. Activate the addition circuitry with the registers used in Steps 1 and 2 as inputs and another register designated to hold the result.

Step 4. Store the result in memory.

Step 5. Stop.



Represented by instructions

Step 1. Get one of the values to be added from memory and place it in a register.

Step 2. Get the other value to be added from memory and place it in another register.

Step 3. Activate the addition circuitry with the registers used in Steps 1 and 2 as inputs and another register designated to hold the result.

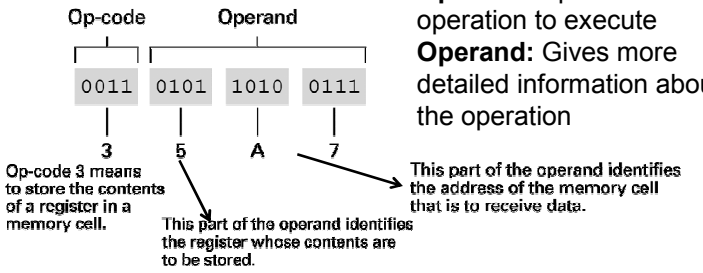
Step 4. Store the result in memory.

Step 5. Stop.

Encoded instructions	Translation
156C	Load register 5 with the bit pattern found in the memory cell at address 6C.
166D	Load register 6 with the bit pattern found in the memory cell at address 6D.
5056	Add the contents of register 5 and 6 as though they were two's complement representation and leave the result in register 0.
306E	Store the contents of register 0 in the memory cell at address 6E.
C000	Halt.

Instruction format

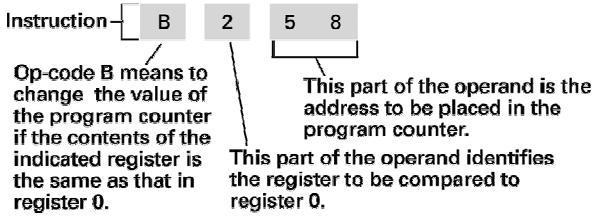
- Store the data in register 5 to memory cell at address A7



Op-code: Specifies which operation to execute
Operand: Gives more detailed information about the operation

Another example

- JUMP to instruction at address 58_H if the content of register 2 is the same as that of register 0



The instruction repertoire

- Which instructions should be included?
- For example, swapping v[k] and v[k+1]

Create a new instruction, called swp, which swaps data in two memory addresses.

```
swp   0($2), 4($2)
```

Complex Instruction Set Computing (**CISC**)

Using load and store instructions

```
lw    $15,  0($2)
lw    $16,  4($2)
sw    $16,  0($2)
sw    $15,  4($2)
```

Reduced Instruction Set Computing (**RISC**)

Instruction types

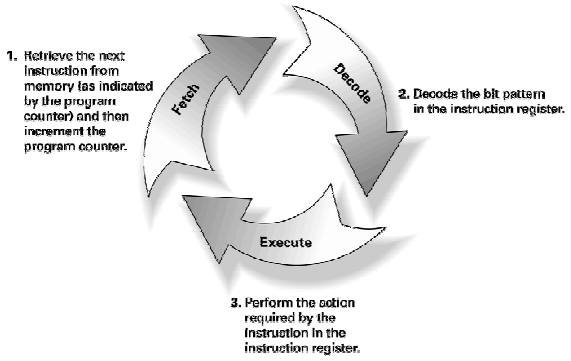
- Data Transfer
  - Copy data between CPU and main memory
  - E.g., LOAD, STORE, device I/O,
- Control
  - Direct the execution of the program
  - E.g., JUMP, BRANCH, JNE (conditional jump),
- Arithmetic/Logic
  - Use existing data values to compute a new value
  - E.g., AND, OR, XOR, SHIFT, ROTATE, etc.

Instruction types

Encoded instructions	Translation	
156C	Load register 5 with the bit pattern found in the memory cell at address 6C.	Data transfer
166D	Load register 6 with the bit pattern found in the memory cell at address 6D.	Data transfer
5056	Add the contents of register 5 and 6 as though they were two's complement representation and leave the result in register 0.	Arithmetic/Logic
306E	Store the contents of register 0 in the memory cell at address 6E.	Data transfer
C000	Halt.	Control

Program Execution

Program execution cycle



How to make a program "run"?

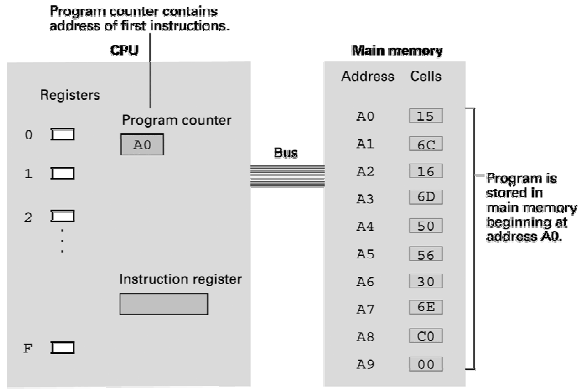
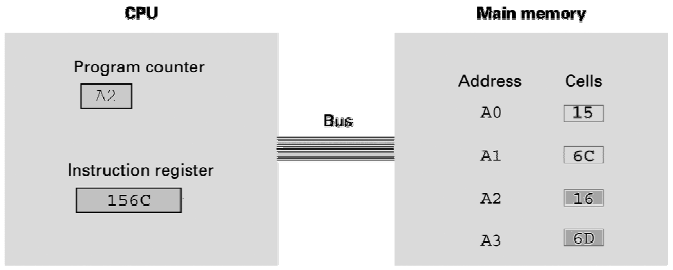


Figure 2.10 The program from Figure 2.7 stored in main memory

Instruction fetch

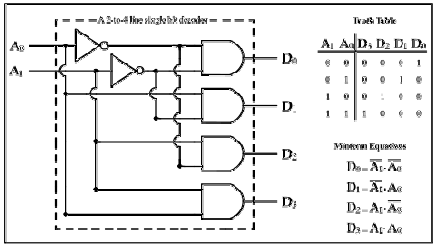


b. Then the program counter is incremented so that it points to the next instruction.

Figure 2.11 Performing the fetch step of the machine cycle

Instruction decode

- How to map opcodes to desired circuits on a CPU?
- For example:
  - 00_b: add
  - 01_b: or
  - 10_b: jump
  - 11_b: and



Interpretation of operand

- The interpretation of operand depends on the op-code

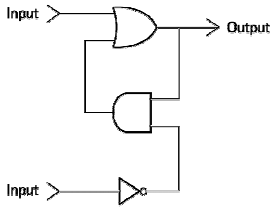
Opcode	Operand	Description
1	4 A 3	**Load** the content at address A3 to register 4
2	4 A 3	**Load** value “A3” to register 4
4	0 A 3	**Move** the content of register A to register 3

Instruction execution

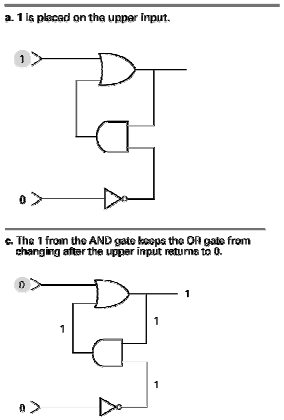
- Uses logic circuits
- Data transfer: load, store, …
  - Logic circuit for registers (Ex: flip-flops)
- Control: jump, jump-equal, …
  - Change the value of *program counter* (PC)
  - Comparison logic circuit
- Arithmetic/Logic: add, and, shift, …
  - Again, logic circuits (adder, as we have seen.)

Flip-flops

- A logic circuit that can store one bit.
  - One input is used to set value 1
  - One input is used to set value 0
  - While both input lines are 0, the most recently stored value is preserved



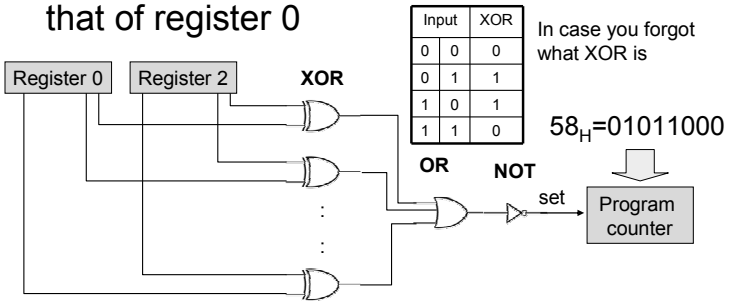
Flip-flops continue



With this, we can set and store value in a register

Example of jump-equal

- B258: JUMP to instruction at address 58_H if the content of register 2 is the same as that of register 0



Input		XOR
0	0	0
0	1	1
1	0	1
1	1	0

In case you forgot what XOR is

58_H=01011000

Exercises

Suppose PC=B0

1. What is in register 3 after the first instruction?
2. What is the memory cell B8 when the program halts?

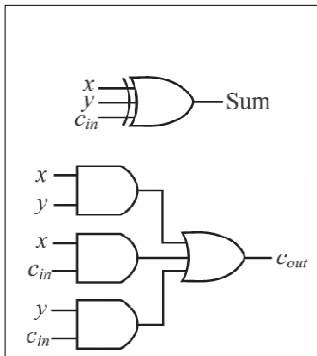
Address	Contents
B0	13
B1	B8
B2	A3
B3	02
B4	33
B5	B8
B6	C0
B7	00
B8	0F

Arithmetic/Logic Operations

Arithmetic/Logic operations

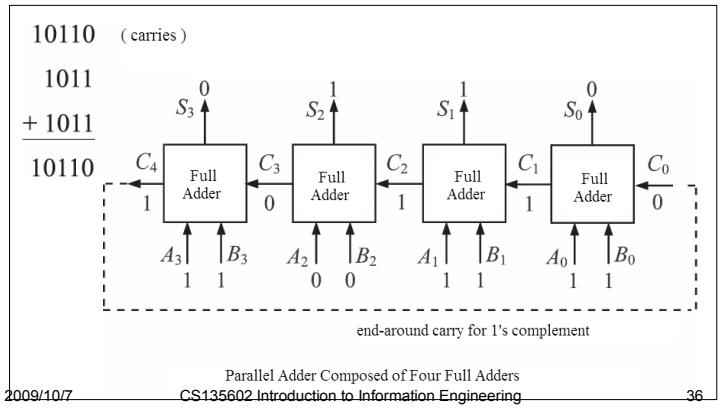
- Arithmetic: add, subtract, multiply, divide
  - Precise action depends on how the values are encoded (two's complement vs. floating-point)
- Shift
  - circular shift (Rotate), logical shift, arithmetic shift
- Logic: AND, OR, XOR, NOT
  - Masking

One bit full adder



	Inputs:			Outputs:	
	Carry in	Addend A	Addend B	Carry Out	Sum
	0	0	0	0	0
	0	0	1	0	1
	0	1	0	0	1
	0	1	1	1	0
	1	0	0	0	1
	1	0	1	1	0
	1	1	0	1	0
	1	1	1	1	1

4 bit parallel adder



Parallel Adder Composed of Four Full Adders

Rotate operation

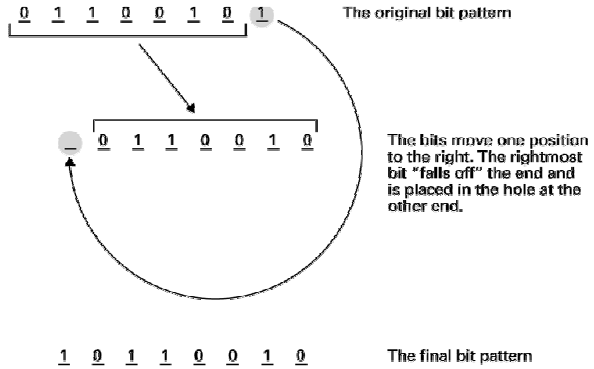


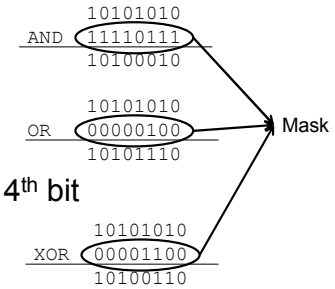
Figure 2.12 Rotating the bit pattern 65 (hexadecimal) one bit to the right

Shift operation

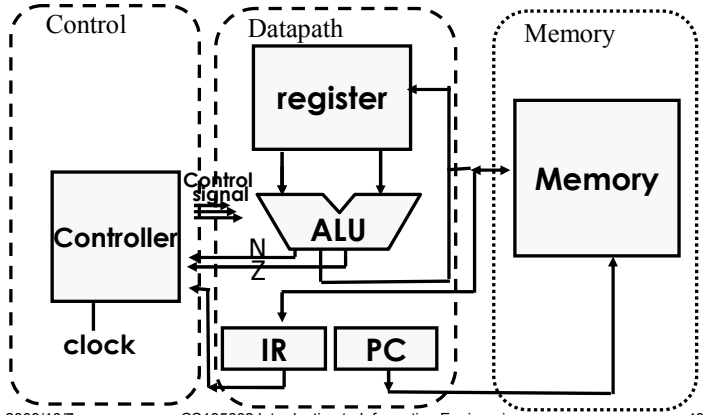
- Circular shift (rotation)
- Logical shift
  - Filling the hole with bit 0
  - Original: 00000101_b ➔ 5_d
  - After 1 left shifting: 00001010_b ➔ 10_d
  - After 2 left shifting: 00010100_b ➔ 20_d
- Arithmetic shift
  - Shifts that leaves the sign bit unchanged

Masking

- AND, OR, XOR can be used for masking
- Example: bit operations on 10101010_b
  - Set the 4th bit to 0
  - Set the 3rd bit to 1
  - Invert the 3rd and the 4th bit



Put everything together



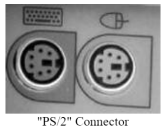
Exercises

- Design a mask to isolate the middle four bits of a byte (set others = 0).
- Encode each of the following commands
  - ROTATE the contents of register 7 to the right 5 bit positions
  - ADD the contents of registers 5 and 6 as thought they were values in floating-point notation and leave the result in register 4
  - AND the contents of registers 5 and 6, leaving the result in register 4.

Peripheral Devices

Connecting to other devices

- Outside the case
  - **Port:** The point at which a device connects to a computer

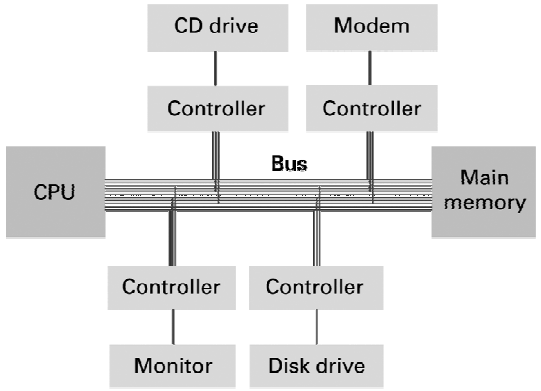

"PS/2" Connector


RS-232 "Serial Port"


USB "A-Style"


IEEE 1394 "Firewire" 6-Pin style

Inside the case

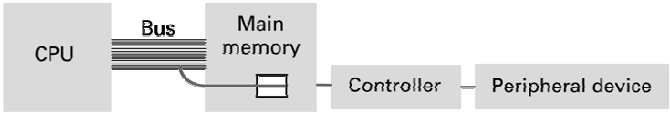


Device controller

- An intermediary apparatus that handles communication between the computer (CPU/memory) and a device.
- Two types of controllers
  - Specialized controllers
    - Network card, graphics card, …
  - General purpose controllers
    - USB, FireWire, …

Device addressing

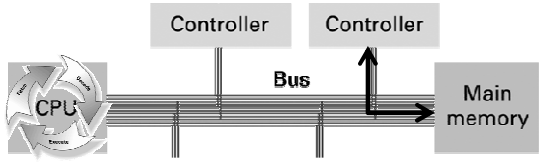
- Memory-mapped I/O:
  - CPU communicates with peripheral devices as though they were memory cells
  - Use load and store to access device data



- Dedicated I/O instructions for devices

Direct memory access (DMA)

- DMA is a mechanism for devices to access memory without occupying CPU.
- At the same time, CPU can execute “other process” until the I/O is finished.
  - Better system throughput



Communication type

- Parallel Communication:
  - Several communication paths transfer bits simultaneously.
  - Printer, computer bus
- Serial Communication:
  - Bits are transferred one after the other over a single communication path.
  - USB, FireWire

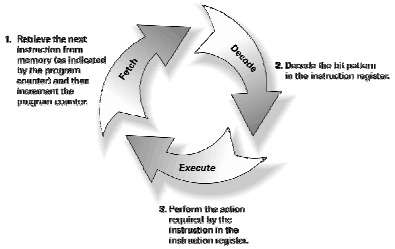
Exercises

- Suppose the machine use memory I/O and the memory address B5 is the location within the printer port to which data to be printed. If register 7 contains the ASCII code for the letter A, what instruction can make letter A to be printed?
- If a printer can only print 128 characters per second, and has local buffer of 256KB, how fast the data rate (bps) can be?

Parallel Architectures

Pipeline

- Program execution is divided into three stages: fetch, decode, execute
  - Suppose each stage takes 3 clock cycles.
  - How many clock cycles are needed to execute 50 instructions?



Pipeline

- Since the hardware used in each stage is separated, CPU can overlap the stages

Fetch instruction 1	Decode instruction 1	Execute instruction 1	Fetch instruction 4	Decode instruction 4
	Fetch instruction 2	Decode instruction 2	Execute instruction 2	Fetch instruction 5
		Fetch instruction 3	Decode instruction 3	Execute instruction 3

- The more stages, the better throughput ?
  - The Pentium 4 had a 35-stage pipeline.

Parallel architectures

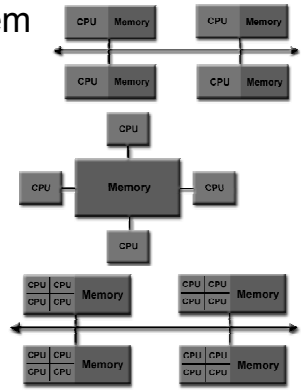
- Bit-level parallelism:
  - 1 bit adder vs. 4 bit adder
- Instruction-level parallelism
  - Pipeline: overlap instruction execution stages
- IO/computation parallelism
  - DMA: overlap communication/computation
- Multiprocessor parallelization
  - Cluster, multi-core processors, GPU

Flynn's taxonomy

- Based on the number of concurrent instruction and data streams available in the architecture (Michael J. Flynn, 1966)
  - SISD (Single-instruction, single-data stream)
    - No parallel processing
  - MIMD (Multiple-instruction, multiple data stream)
    - Different programs, different data
  - SIMD: (Single instruction, multiple data stream)
    - Same program, different data

By memory location

- Distributed memory system
  - Multiple processors that communicate through a computer network.
- Shared memory system
  - Multiple processors that communicate through a shared memory space.
- Hybrid system



Speedup

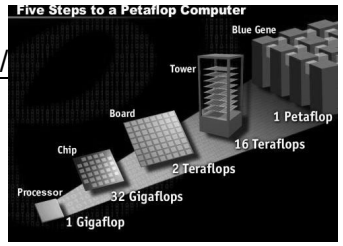
$$S(n) = \frac{\text{Execution time using one processor (single processor system)}}{\text{Execution time using a multiprocessor with } n \text{ processors}} = \frac{t_s}{t_p}$$

- Amdahl's law
  - Suppose there are *f*% of tasks cannot be parallelized, the best speedup by n processors is

$$S(n) = \frac{t_s}{ft_s + (1-f)t_s/n} = \frac{n}{1+(n-1)f}$$

Supercomputers

- Hundred thousands of processors interconnected via special designed network
  - Top1: Roadrunner
  - http://www.top500.org/



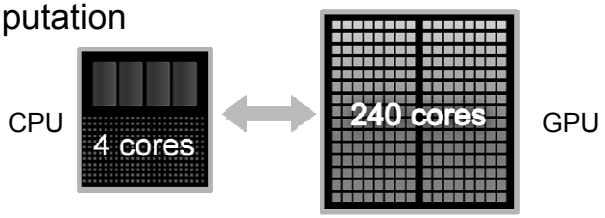
Multi-core processor

- A processor composed of two or more independent cores (or CPUs).
- Advantages
  - Performance improvement
  - Low power consumption
- Disadvantages
  - Operating system support
  - Software support

We will talk those problems later

Graphics processing unit (GPU)

- A specialized processor designed for 3D graphics rendering
- Modern GPU has over thousand cores, which can be used for general purpose computation



Exercises

- Suppose instructions can be fully overlapped in a 3 stages pipeline CPU, and each stage takes 3 clock cycles, how many clock cycles are needed to execute 500 instructions? How if there are 5 stages?
- What is the best speedup for 10 processors if there are 20% of tasks can be parallelized? How about 60%?

Related courses

- Store program concept, peripheral devices
  - 計算機結構，硬體實驗，微算機系統，邏輯設計，嵌入式系統概論
- Parallel Architectures

References

- http://www.top500.org/ (supercomputer)
- https://computing.llnl.gov/tutorials/parallel_comp/
- www.cs.nthu.edu.tw/~ychung/slides/para_programming/slides1.pdf
- Textbook chap 2

Opcode	Operand	Description
1	RXY	LOAD the register R with the bit pattern found in the memory cell whose address is XY. *Example:* I4A3 would cause the contents of the memory cell located at address A3 to be placed in register 4.
2	RXY	LOAD the register R with the bit pattern XY. *Example:* 20A3 would cause the value A3 to be placed in register 0.
3	RST	STORE the bit pattern found in register R in the memory cell whose address is XY. *Example:* 35B1 would cause the contents of register 5 to be placed in the memory cell whose address is B1.
4	ORS	MOVE the bit pattern found in register R to register S. *Example:* 40A4 would cause the contents of register A to be copied into register 4.
5	RST	ADD the bit patterns in registers S and T as though they were two's complement representations and leave the result in register R. *Example:* 5726 would cause the binary values in registers 2 and 6 to be added and the sum placed in register 7.

Opcode	Operand	Description
6	RST	ADD the bit patterns in registers S and T as though they represented values in floating point notation and leave the floating-point result in register R. *Example:* 634E would cause the values in registers 4 and E to be added as floating-point values and the result to be placed in register 3.
7	RST	OR the bit patterns in registers S and T and place the result in register R. *Example:* 7CB4 would cause the result of ORing the contents of registers Band 4 to be placed in register C.
8	RST	AND the bit patterns in registers S and T and place the result in register R. *Example:* 8045 would cause the result of ANDing the contents of registers 4 and 5 to be placed in register 0.
9	RST	EXCLUSIVE OR the bit patterns in registers Sand T and place the result in register R. *Example:* 95F3 would cause the result of EXCLUSIVE ORing the contents of registers F and 3 to be placed in register 5

Opcode	Operand	Description
A	R0X	ROTATE the bit pattern in register R one bit to the right X times. Each time place the bit that started at the low-order end at the high-order end. *Example:* A403 would cause the contents of register 4 to be rotated 3 bits to the right in a circular fashion.
B	RXY	JUMP to the instruction located in the memory cell at address XY if the bit pattern in register R is equal to the bit pattern in register number 0. Otherwise, continue with the normal sequence of execution. (The jump is implemented by copying XY into the PC during the execute phase.) *Example:* B43C would first compare the contents of register 4 with the contents of register 0. If the two were equal, the pattern 3C would be placed in the program counter so that the next instruction executed would be the one located at that memory address. Otherwise, nothing would be done and program execution would continue in its normal sequence.
C	000	HALT execution. *Example:* C000 would cause program execution to stop.