Data Structures and Algorithms

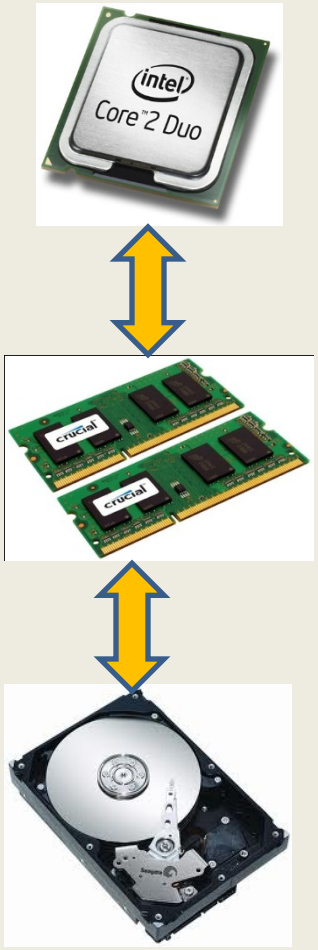
(ESO207)

Lecture 2:

- Model of computation
- Efficient algorithm for **F**(n) **mod** m.

RECAP OF THE 1ST LECTURE

Current-state-of-the-art computer



A processor (CPU)

speed = few GHz

(a few **nanoseconds** to execute an instruction)

Internal memory (RAM)

size = a few GB (Stores few million bytes/words)

speed = a few GHz(a few **nanoseconds** to read a byte/word)

External Memory (Hard Disk Drive)

size = a few tera bytes

speed : seek time = **miliseconds**

transfer rate= a **billion** bytes per second

Motivation

- for Efficient algorithms
  - Subset sum problem
  - Sorting
- for Efficient Data Structures
  - Telephone Directory
  - Range Minima

Homework 1

(compulsory)

Write a **C** program for the following problem:

Input: a number n

n : **long long int** (64 bit integer).

Output: **F(n) mod 2014**

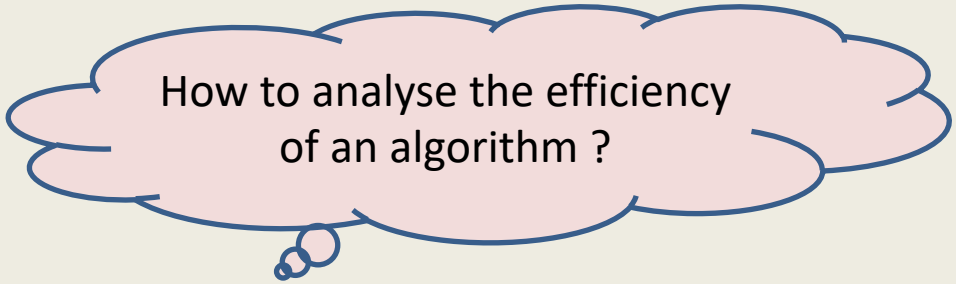
Time Taken	Largest n for Rfib	Largest n for IFib
1 minute	48	4.5×10^9
10 minutes	53	4.5×10^{10}
60 minutes	58	2.6×10^{11}

Processor: 2.7 GHz

Here are the values obtained by executing the program on a typical computer.

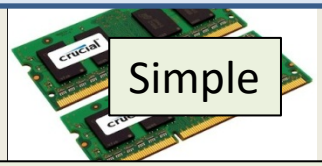
Inferences
from the Homework

- Inference for **RFib** algorithm
  - Too slow ☹
- Inference for **Ifib** algorithm :
  - Faster than **Rfib**
  - But not a solution for the problem ☹
- Efficiency of an algorithm does matter
- Time taken to solve a problem may vary depending upon the algorithm used



Current-state-of-the-art Computer

A model of computation



Simple

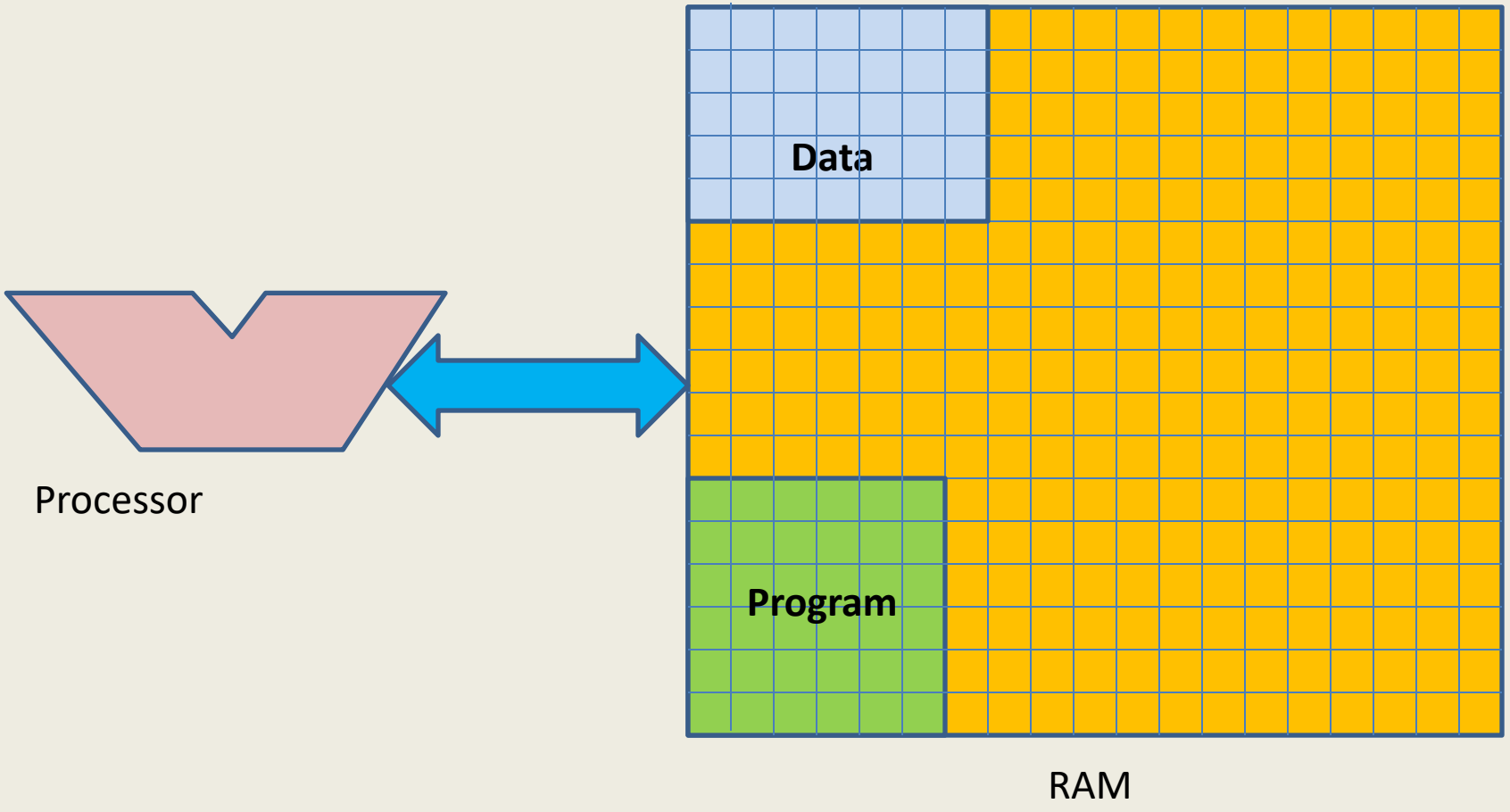
Close to real world computer

word RAM :

a model of computation

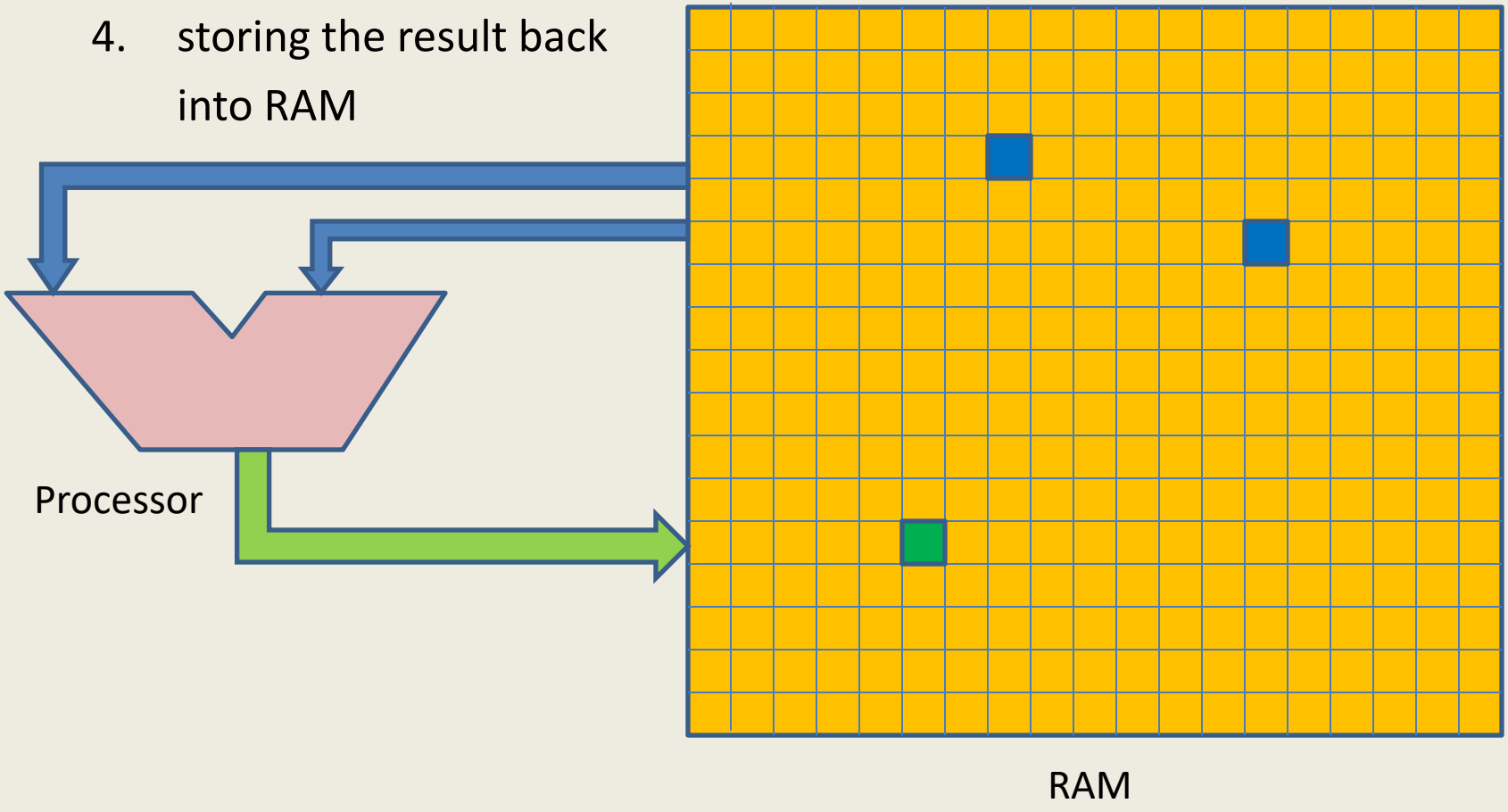
word RAM :

a model of computation



How is an instruction executed?

1. Decoding instruction
2. fetching the operands
3. performing arithmetic/logical operation
4. storing the result back into RAM



➔ Each instruction takes a few cycles (click ticks) to get executed.

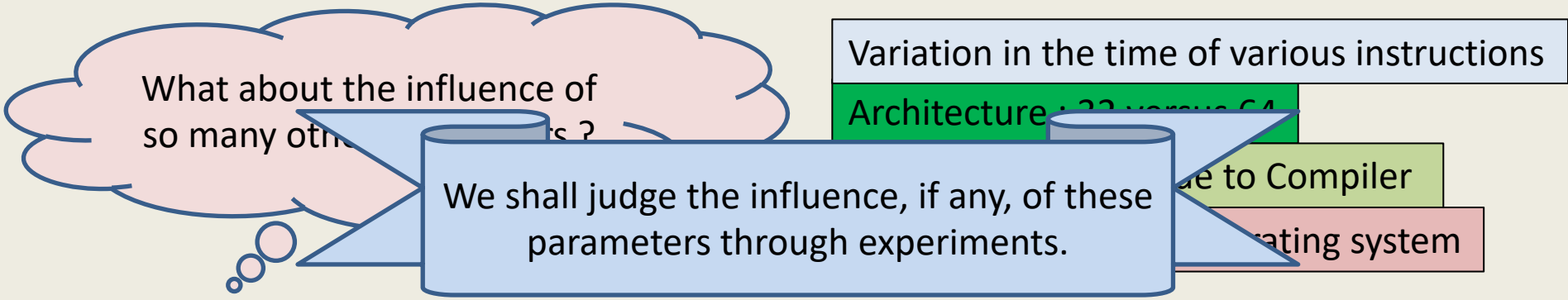
word RAM model of computation: Characteristics

- Word is the **basic storage** unit of RAM. Word is a collection of few bytes.
- Each input item (number, name) is stored in **binary format**.
- RAM can be viewed as a huge array of words.
- Any arbitrary location of RAM can be **accessed** in the same time **irrespective** of the location.
- Data as well as Program **reside fully** in RAM.
- Each arithmetic or logical operation (+,-,*,/,or, xor,…) involving a constant number of words takes **a constant number of cycles (steps)** by the CPU.

Efficiency of an algorithm

Question: How to measure time taken by an algorithm ?

- Number of instructions taken in **word RAM** model.



Who knows, these factors might have little or negligible impact on most of algorithms. ☺

Homework 1 from Lecture 1

Computing F(n) mod m

Iterative Algorithm for F(n) mod m

Total number of instructions=
$4+3(n-1)+1 \approx 3n$

IFib(n,m)

```
if n = 0  return  0;
    else if n =1 return 1;
        else {      a ← 0;  b← 1;
                    For(i =2 to n) do
                    {    temp ← b;
                         b← a + b  mod m ;
                         a ← temp;
                    }
             }
  return b;
```

4 instructions

n-1 iterations

3 instructions per iteration

the final instruction

Let us calculate the number of instructions executed by IFib(n,m)

Recursive algorithm for F(n) mod m

RFib(n,m)

{ if n =0 return 0;

else if n =1 return 1;

else return((**RFib**$(n-1,m)$ + **RFib**$(n-2,m)$) mod m)

}

Let G(n) denote the number of instructions executed by RFib(n,m)

- **G(0) = 1;**
- **G(1) = 2;**
- For n **>1**

$$\mathbf{G}(n) = \mathbf{G}(n-1) + \mathbf{G}(n-2) + 4$$

Observation 1: **G**(n) >**F**(n) for all n;

➔**G**(n)> $2^{(n-2)/2}$**!!!**

Algorithms for F(n)mod m

- # instructions by **Recursive** algorithm **RFib**(n): $> 2^{\frac{n-2}{2}}$

  (exponential in n)

- # instructions by **Iterative** algorithm **IFib**(n): $3n$

  (linear in n)

None of them works for entire range of **long long int** n and **int** m

Question: Can we compute **F**(n)**mod** m quickly ?

How to compute F(n)mod m quickly ?

… need some better insight …

A warm-up example

How good are your programming skills ?

Compute $x^n \bmod m$

Problem: Given three integers x, n, and m, compute $x^n \bmod m$.

$$x^n = \begin{cases} 1 & \text{if } n = 0 \\ x \times x^{n-1} & \text{otherwise} \end{cases}$$

Power(x, n, m)

```
If (n = 0) return 1;
 else {
        temp ← Power(x, n − 1, m);
        temp ← (temp × x) mod m ;
        return temp;
      }
```

4 instructions excluding the **Recursive** call

Compute $x^n \bmod m$

Problem: Given three integers x, n, and m, compute $x^n \bmod m$.

$$x^n = \begin{cases} 1 & \text{if } n = 0 \\ x \times x^{n-1} & \text{otherwise} \end{cases}$$

Power(x, n, m)

Power$(x, n-1, m)$;

Power$(x, n-2, m)$;

Power$(x, 0, m)$

No. of instructions executed by **Power**(x, n, m) = $4n$

Compute $x^n \bmod m$

Problem: Given three integers x, n, and m, compute $x^n \bmod m$.

$$x^n = \begin{cases} 1 & \text{if } n = 0 \\ x^{n/2} \times x^{n/2} & \text{if } n \text{ is even} \\ x^{n/2} \times x^{n/2} \times x & \text{if } n \text{ is odd} \end{cases}$$

Power(x, n, m)

```
If (n = 0) return 1;
else {
        temp ← Power(x, n/2, m);
        temp ← (temp × temp) mod m ;
        if (n mod 2 = 1 ) temp ←(temp × x)mod m ;
       return temp;
      }
```

5 instructions excluding the **Recursive** call

Compute $x^n \bmod m$

Problem: Given three integers x, n, and m, compute $x^n \bmod m$.

$$x^n = \begin{cases} 1 & \text{if } n = 0 \\ x^{n/2} \times x^{n/2} & \text{if } n \text{ is } \textbf{even} \\ x^{n/2} \times x^{n/2} \times x & \text{if } n \text{ is } \textbf{odd} \end{cases}$$

Power(x, n, m)

Power$(x, n/2, m)$

Power$(x, n/4, m)$

Power$(x, 0, m)$

No. of instructions executed by **Power**(x, n, m) = $5 \log_2 n$

Efficient Algorithm for F(n)mod m

Idea 1

Question: Can we express **F**(n) as a^n for some constant a ?

Unfortunately **no**.

Idea 2

$$\begin{pmatrix} F(n) \\ F(n-1) \end{pmatrix} = \begin{pmatrix} 1 & ? & 1 \\ 1 & & 0 \end{pmatrix} \times \begin{pmatrix} F(n-1) \\ F(n-2) \end{pmatrix}$$

Unfolding the RHS of this equation, we get …

$$\begin{pmatrix} F(n) \\ F(n-1) \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^{n-1} \times \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

A clever algorithm for F(n)mod m

Clever-algo-Fib(n, m)

{ $A \leftarrow \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$;

$B \leftarrow A^{n-1} \bmod m$;

$C \leftarrow B \times \begin{pmatrix} 1 \\ 0 \end{pmatrix}$;

return $C[1]$; // the first element of vector C stores F(n)**mod** m

}

4 instructions

6 instructions

Question: How to compute $\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^{n-1}$ efficiently ?

Answer :

Inspiration from Algorithm for x^n **mod** m

A clever algorithm for F(n)mod m

Let A be a 2×2 matrix.

- If n is even, $A^n = A^{n/2} \times A^{n/2}$
- If n is odd, $A^n = A^{n/2} \times A^{n/2} \times A$

Question: How many instructions are required to multiply two 2×2 matrices ?

Answer: 12

Question: Number of instructions for computing $\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^{n-1} \bmod m$?

Answer : 35 $\log_2 (n-1)$

Question: Number of instructions in **New-algo-Fib**(n, m)

Answer: 35 $\log_2(n-1)$+ 11

Three algorithms

Algorithm for F(n)mod m	No. of Instructions
RFib(n,m)	$> 2^{(n-2)/2}$
IterFib(n,m)	$3n$
Clever_Algo_Fib(n,m)	$35 \log_2 (n-1) + 11$

?

- Which algorithm is the best ?
- What is the exact base of the exponent in the running time of **RFib** ?
- Are we justified in ignoring the influence of so many other parameters ?
  (Variation in the time of instructions/Architecture/Code optimization/…)
- How close to **reality** is the RAM model of computation ?

Find out yourself !