ECE 131 – Programming Fundamentals

Module 1, Lecture 2: Historical Background

Dr. Daryl Lee

University of New Mexico

A Brief History of Computing

- What is an algorithm? A step by step set of instructions for solving a given problem.
- Is this an algorithm?
  - mix 2 cups sifted flour, 2 cups brown sugar and 1/2 cup butter
  - spread mixture evenly in a greased 8-inch square pan
  - bake at 350° for 35–40 min.
  - cool and dust with powdered sugar

  Yes, it's an algorithm for baking a cake.
- The idea of an algorithm is quite old. Indeed the oldest algorithm is attributed to the Greek mathematician Euclid, who specified an algorithm for finding the greatest common divisor (gcd) of two integers around 300 BC. The gcd of two integers x and y, denoted $\gcd(x, y)$, is the largest integer that divides both x and y.
  Ex: $\gcd(80, 32) = 16$.

THE UNIVERSITY of NEW MEXICO

A Brief History of Computing

- How would you compute the Greatest Common Denominator of two numbers?
- Example: 87, 18

  $18/9 = 2; 87/9 \tilde{=} 9.7$ XXX
  $18/8 = 2.25$ XXX
  $18/7 \tilde{=} 2.6$ XXX
  $18/6 = 3; 87/6 \tilde{=} 14.5$ XXX
  $18/5 = 3.6$ XXX
  $18/4 = 4.5$ XXX
  $18/3 = 6; 87/3 = 29;$!!!

- After 7 tries and 10 divisions we found the GCD(87,18) = 3.

THE UNIVERSITY of NEW MEXICO

A Brief History of Computing

Here's Euclid's algorithm:

EUCLID(integer a, integer b)

```
1    while b ≠ 0 do
2        temp ← b
3        b ← a mod b
4        a ← temp
5    return a
```

Euclid.

Ex: EUCLID$(87, 18) \Rightarrow a \leftarrow 18,\ b \leftarrow 15$
$\Rightarrow a \leftarrow 15,\ b \leftarrow 3$
$\Rightarrow a \leftarrow 3,\ b \leftarrow 0$
return 3

3 iterations, 3 divides.

THE UNIVERSITY of NEW MEXICO

A Brief History of Computing

Here's an edge condition:

EUCLID(integer a, integer b)

```
1    while b ≠ 0 do
2        temp ← b
3        b ← a mod b
4        a ← temp
5    return a
```

Euclid.

Ex: EUCLID$(97, 18) \Rightarrow a \leftarrow 18,\ b \leftarrow 7$
$\Rightarrow a \leftarrow 7,\ b \leftarrow 4$
$\Rightarrow a \leftarrow 4,\ b \leftarrow 3$
$\Rightarrow a \leftarrow 3,\ b \leftarrow 1$
$\Rightarrow a \leftarrow 1,\ b \leftarrow 0$
return 1

Since the gcd of 97 and 18 is 1, they are relatively prime.

THE UNIVERSITY of NEW MEXICO

A Brief History of Computing

- In 825 AD, the Persian astronomer and mathematician Mohammed al-Khowârizmî specified algorithms for adding, subtracting, multiplying and dividing decimal numbers.
- In the 12th century, this work was translated into Latin, with the Latin translation of the author's name being "Algoritmi".
- The eventually led to the word "algorithm", with the common meaning "calculation method".
- During the 1800's, a number of mechanical machines were built that could be said to implement algorithms.
  - In 1801, Joseph Jacquard invented a weaving loom where the woven pattern was determined by cards with holes punched at various locations.
  - Charles Babbage attempted to build a machine in 1833, which he called the "difference engine", for evaluating certain mathematical formulas.
  - Herman Hollerith invented a machine, that also used punched cards, that was used by the American Census Bureau to help tabulate the national census in 1890.

A Brief History of Computing

- During the 1930's, mainly due to war efforts associated with breaking cryptographic codes and calculating the trajectories of projectiles, scientists began to explore the notion of building general purpose electronic computers.
- Alan Turing, a mathematician, formalized the concept of algorithms, and their implementation as programs that could execute on a general-purpose computing device.

Alan Turing.

THE UNIVERSITY of NEW MEXICO

A Brief History of Computing

- The honor of being the first general-purpose electronic computer is usually awarded to the ENIAC (Electronic Numerical Integrator and Calculator).
- The ENIAC was built by John Mauchly and J. Presper Eckert between 1943-45 at the University of Pennsylvania.

Mauchly (left) and Eckert (right).

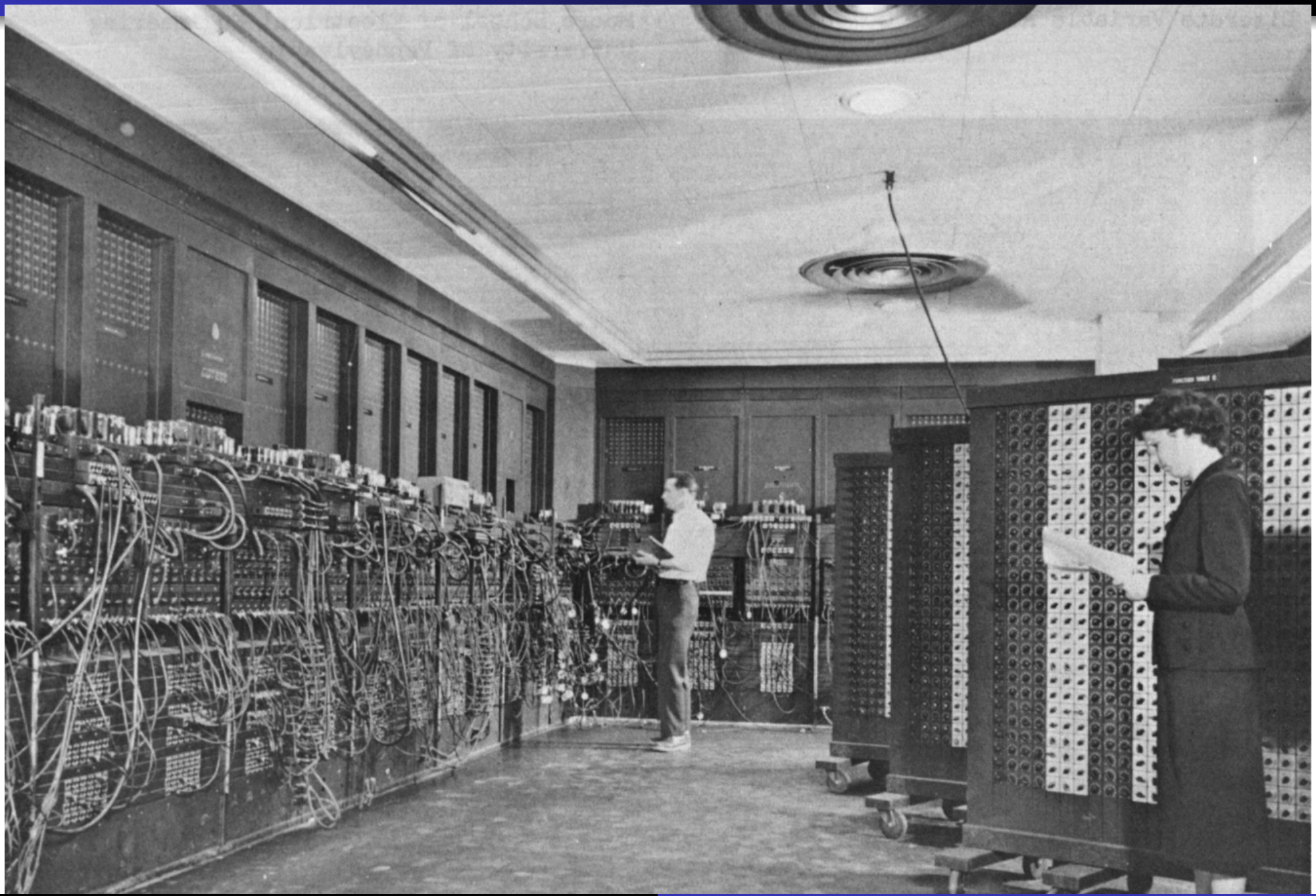
THE UNIVERSITY of NEW MEXICO

A Brief History of Computing

Facts about the ENIAC:

- Filled a 20×40 foot room, weighed 30 tons, and was composed of more than 18,000 vacuum tubes.
- The vacuum tubes generated so much heat (174,000 watts) that heavy duty air conditioning equipment was required to cool the room.
- The ENIAC was funded by the Army in support of the war effort.
- Prior to the ENIAC, the term "computers" was used to refer to the people who were employed by the Army to calculate firing tables for artillery guns.
- Because the ENIAC subsumed this functionality, it became known as a "computer".
- It had a clock speed of 100,000 cycles per second.

The ENIAC



The ENIAC

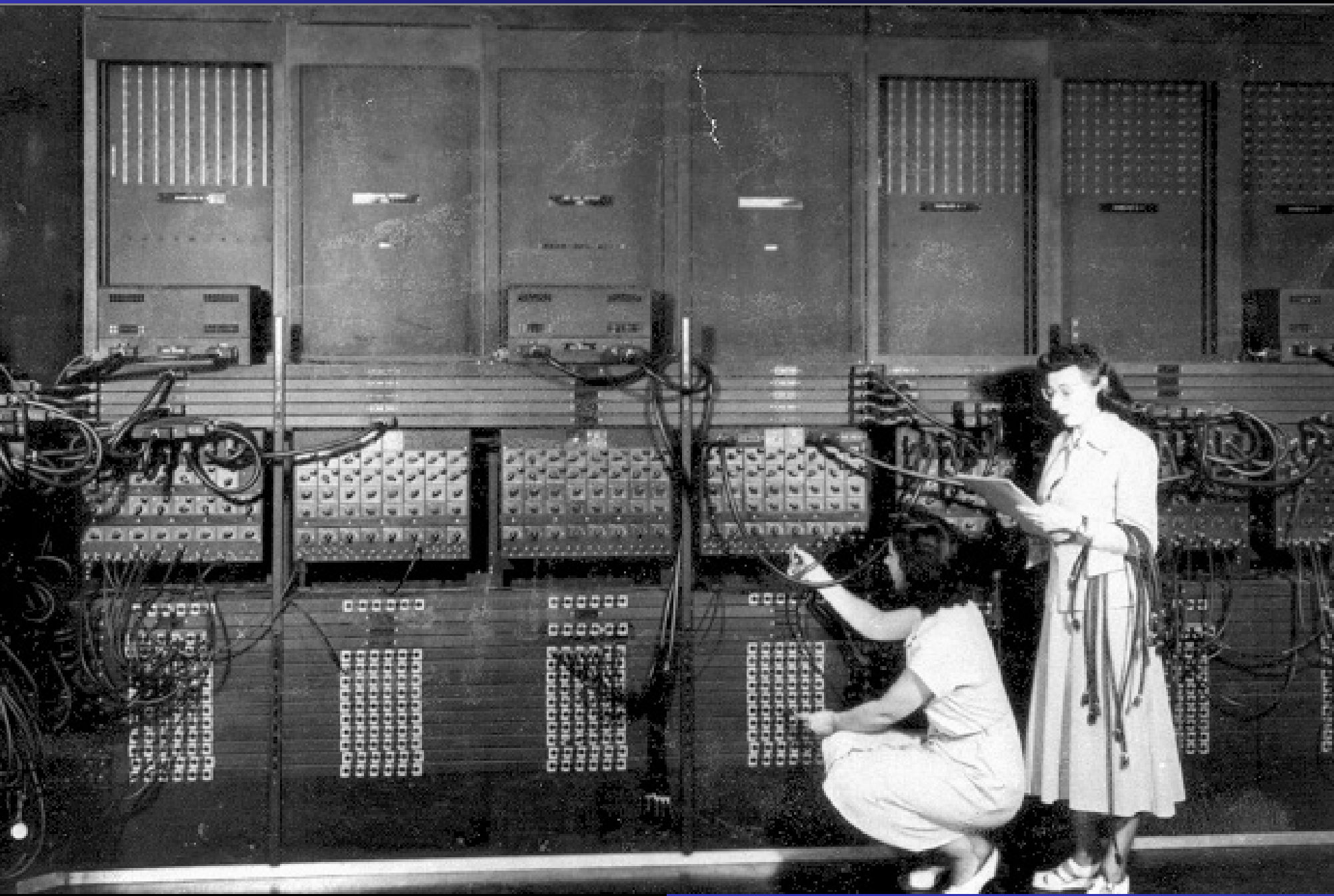
- Vacuum tubes were notoriously unreliable, and the device that had previously used the largest number of them was an electronic organ that used 160 of them.
- For this reason, many faculty refused to join the ENIAC project, as did RCA, the largest manufacturer of vacuum tubes (although they did supply the tubes in the interest of "wartime cooperation").
- Eckert solved the tube reliability problem through an extremely careful circuit design.
- Programming the ENIAC required a physical modification of all its patch cords and switches — a task that could take days.
  Ex: A program statement in C looks like:

  ```
  circumference = 3.14 * radius;
  ```

  To program this same statement on the ENIAC would require moving many patch cords and setting three particular knobs to 3, 1 and 4.
- Because changing a program required the computer to be "re-wired (redesigned if you will), the ENIAC is called a fixed-program computer.

THE UNIVERSITY of NEW MEXICO

The ENIAC



A Brief History of Computing

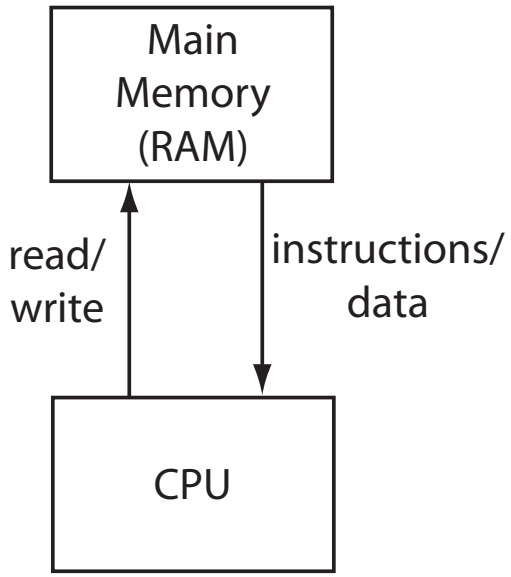
- Eckert and Mauchly teamed with Mathematician John von Neumann to create the EDVAC computer, which implemented the concept of a stored-program computer. I.e., the program is stored in computer memory, along with the data that it operates on.
- Thus, it was no longer required to program a computer using patch cords.

John von Neumann.

THE UNIVERSITY of NEW MEXICO

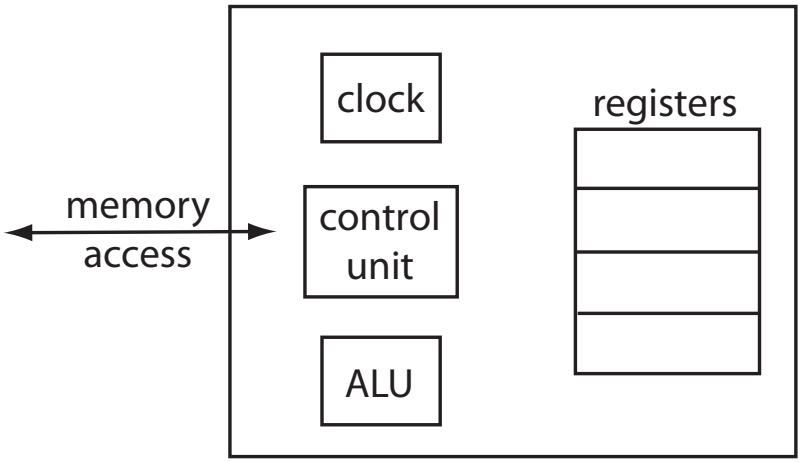
The von Neumann Architecture

This general approach became known as the von Neumann architecture.



- Central Processing Unit (CPU) – executes operations according to the computer's instruction set.
- Main Memory – program instructions and data reside in read/write random access memory.

The CPU – What's Inside

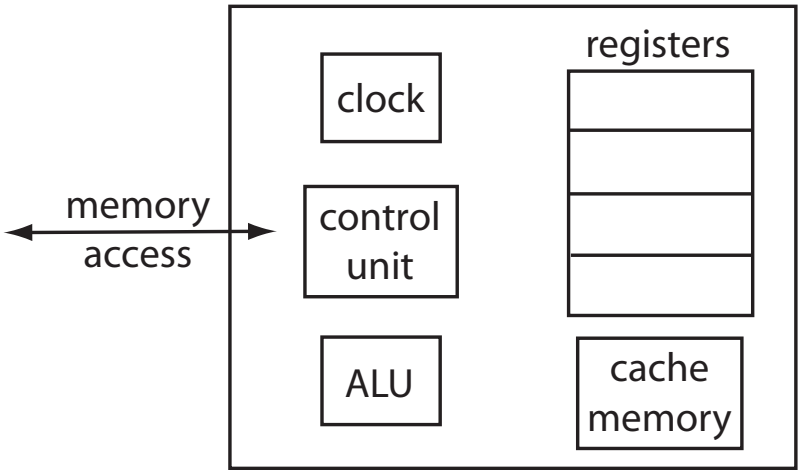


- Control unit – decodes the instruction according to a fetch-execute cycle.
- Arithmetic Logic Unit (ALU) – performs all mathematical operations.
- Registers – small local storage used to store data about to be manipulated.

The von Neurmann Bottleneck

The von Neumann bottleneck –

- In most modern computers, throughput is much smaller than the rate at which the CPU can work.
- Thus, for some memory-intensive programs, the CPU is continuously forced to wait for needed data to be transferred to or from memory.
- Cache memory alleviates this problem somewhat.



THE UNIVERSITY of NEW MEXICO

Machine Language

- Computers were (and still are) built using digital circuits whose inputs and outputs can have one of only two values.
- The values are true (high voltage) or false (low voltage), and are typically represented as 1 and 0, respectively.
- In order to program early computers, one had to directly set the values of these circuits using this binary reprsentation.
- A program for Euclid's algorithm might look something like:

```
00110011
11111010
11110011
01110100
10000011
   ...
```

- Such a program is said to be written in machine code, and the lines of code are called operation codes as they specify a particular instruction (or operation) in a CPU's instruction set.
- The term "operation code" is often shortened to opcode.

THE UNIVERSITY of NEW MEXICO

Assembly Language

Assembly language is a low-level language for programming computers. It implements a symbolic representation of the numeric machine codes and other constants needed to program a particular CPU architecture.

- Assembly language is mnemonic $\Rightarrow$ less prone to make programming errors.
- The following program stores the value 9 (hexadecimal) in register `ah`, and the value 5 in register `al`. It then adds these two values, leaving the result in register `al`.

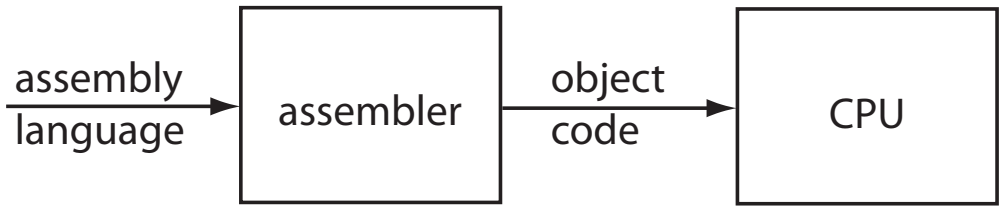
```
mov  ah, 09h
mov  al, 05h
add  al, ah
```

  (You'll see more of this in ECE 344.)
- This is not portable to different CPU architectures – it's machine dependent!

THE UNIVERSITY of NEW MEXICO

Assembly Language

- An assembler is a program that translates assembly language instructions into opcodes.
- It's a program that creates another program that is executable on a given target CPU.
- One line of assembly language code usually translates to one opcode.
- The output of an assembler is typically a file containing the opcodes, and the entire file is referred to as object code:

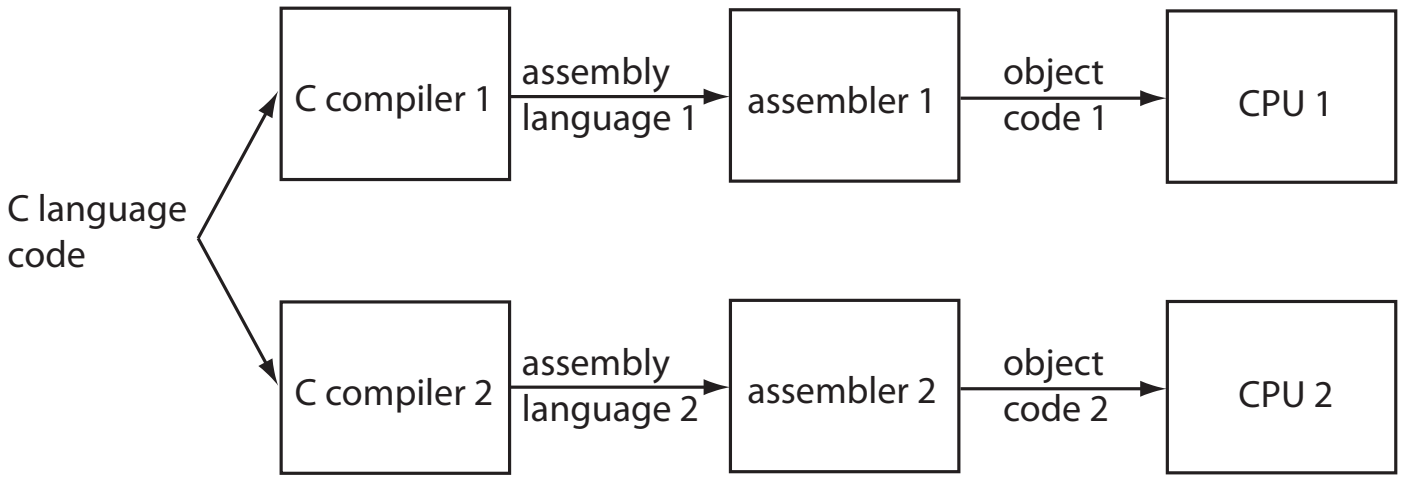


- Because assembly language code is mnemonic, it is said to be more abstract than machine code, and we say that assembly language is at a higher level of abstraction than machine code.

THE UNIVERSITY of NEW MEXICO

High-Level Language

- High level languages (HLLS) are at an even higher level of abstraction than assembly language.
- In a HLL, the details of the target CPU architecture are completely abstracted away (absent).
- Thus, HLLs are portable to different CPU architectures. A compiler is a program that translates code in a HLL into assembly language code:



- One line of code in a HLL can correspond to many lines of assembly language code, and the assembler itself is typically rolled into the compiler.

THE UNIVERSITY of NEW MEXICO

High-Level Language

- FORTRAN (FORmula TRANslation) was a dominant HLL in the early days of computing. Others now include C, C++, Java, Ruby, etc., etc., etc.
- Because of the higher level of abstraction, it is usually easier to solve a problem by coding the solution in a HLL, rather than assembly language – unless highly optimized code is required.
- Over the years compilers have become better and better at producing optimized code, and now it's not very common for programmers to write assembly language code.
- As an engineer, however, there are still occasions when you will need to code in assembly language:
  - When writing device drivers.
  - When working with embedded systems – these often have severe resource constraints.
  - When using CPU specific instructions not supported by a compiler.
  - When reverse engineering code – e.g., your trying to understand what a computer virus does.

THE UNIVERSITY of NEW MEXICO