CS321

I. Programming Languages

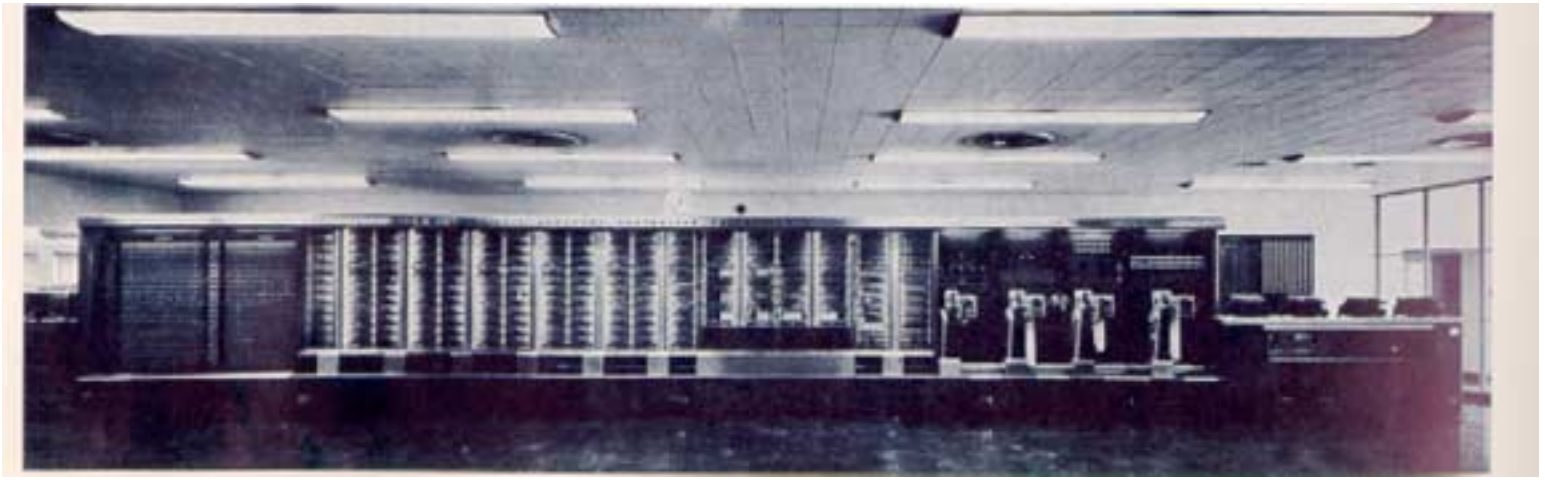
Introduction

- Computers, today can be programmed using a wide range of languages that differ in their syntax, the programming model they implement, how much operational detail is needed in programs written in these languages (level of abstraction), the implementation strategy (translation or interpretation), etc.
- Studying the fundamental concepts underlying the design and implementation of the most important programming languages and is the objective of this course.

Programming of early computers

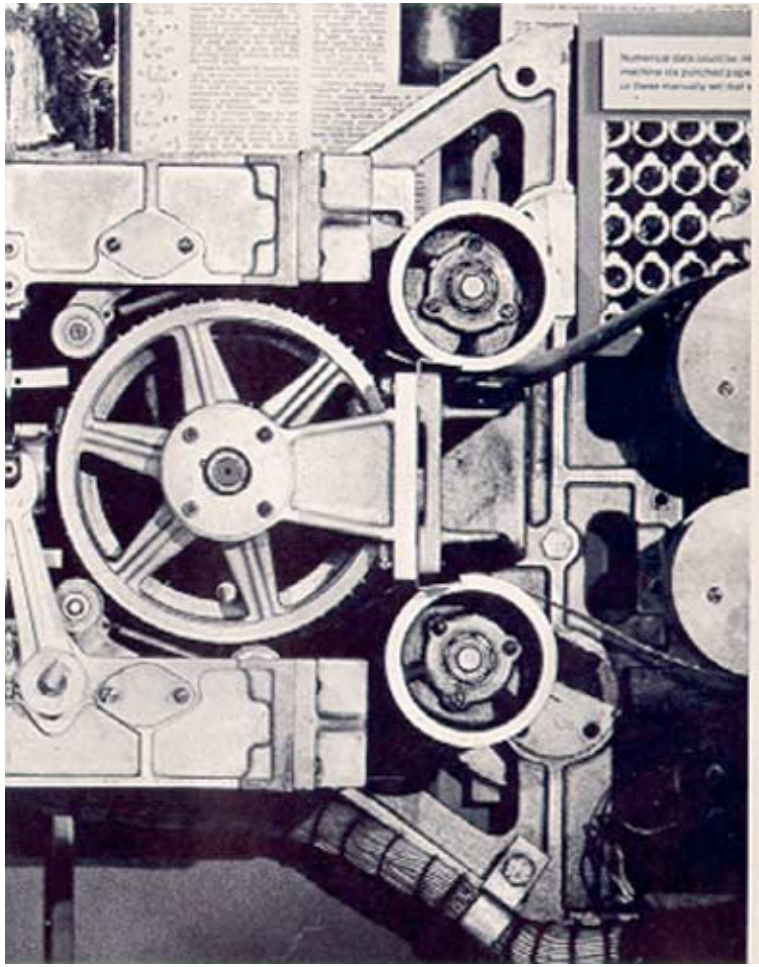
- Early computers were programmed directly in machine language.
  - There were no software tools.
  - Limited resources and speed required very efficient programming.
- The Automatic Sequence Controlled Calculator (Mark I)
  - Harvard. 1944.
  - Mostly mechanical machine.
  - Programs resided on paper tape.

- There were 24 positions across the tape in which holed could be punched. Each position represented a bit.
- Each row of holes contained an instruction.
- When each instruction has been performed the tape is stepped on so that the next row of holes can be read.
- Each instruction contained three 8-bit codes, a source address, a destination address, and an operation code.
- The tape was punched with the aid of a keyboard perforator having three groups of eight keys; each key punches one hole on the tape.



Harvard graduate student Howard H. Aiken proposed, in 1937, that a new kind of calculating machine be built. He wrote: there exist problems beyond our ability to solve, not because of theoretical difficulties, but because of insuficient means of mehcanical computation."

Aiken was thinking of linking togethher Monroe calculators on the player piano principle to create a large scientific calculator, but Harvard astronomer Harlow Shapley sent him to IBM.

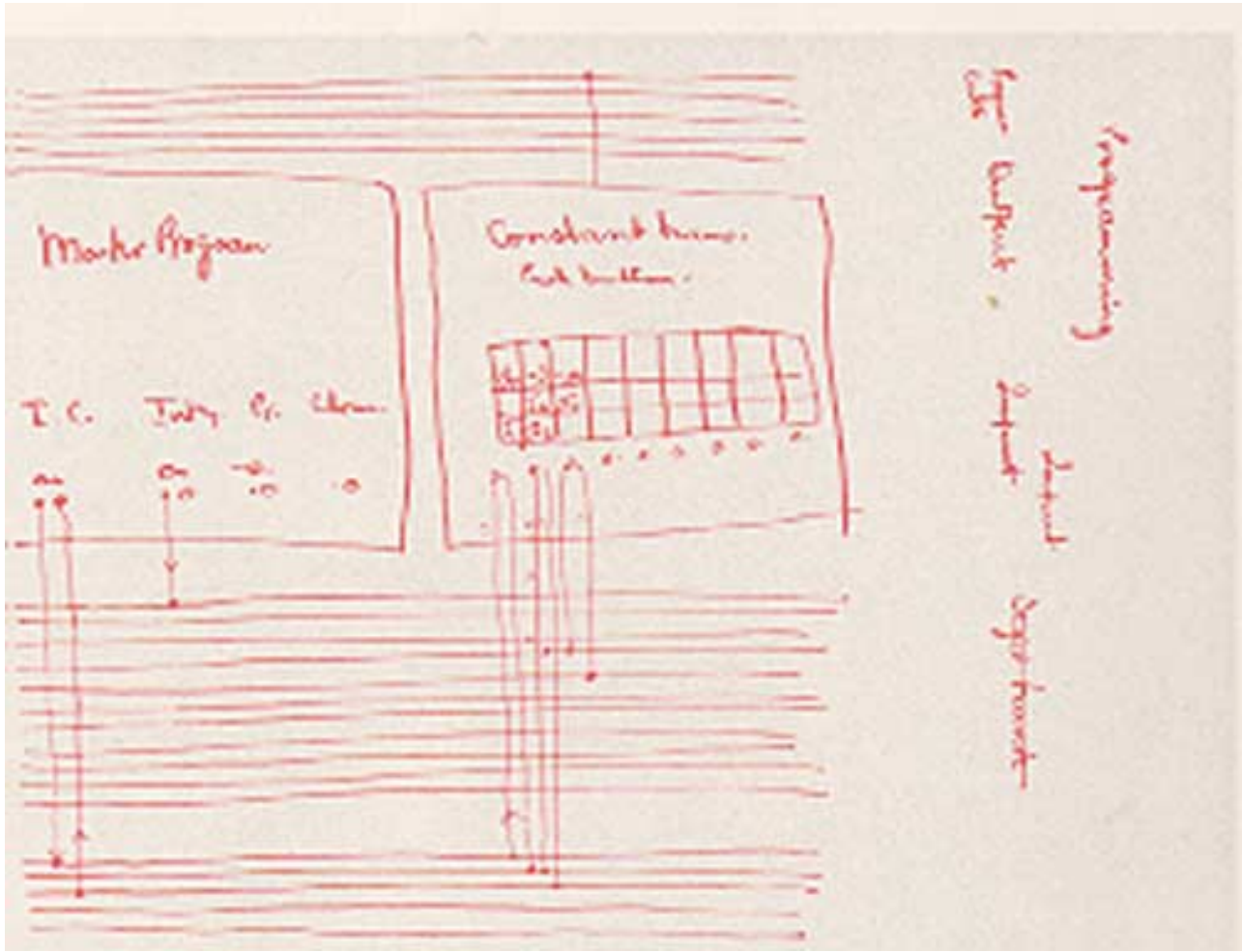


- The Electronic Numerical Integrator and Computer (ENIAC)
  - Moore School of Electrical Engineering. University of Pennsylvania. 1946
  - Vacuum tubes.
  - “Putting a program on the ENIAC involves making a large number of connexions with plugs and sockets and setting up a large number of switches” [M.V.Wilkes. Automatic Digital Computers]

Coinventors J. Presper Eckert and John Mauchly in the foregorund

Herman Goldstine and J. Presper Eckert

Adele Goldstine, the first programmer of the ENIAC, and her husband, Captain Herman H. Goldstine



- The Electronic Delay Storage Automatic Calculator (EDSAC)
  - Mathematical Laboratory. Cambridge University. 1949.
  - A binary store-program computer.
  - Had a rudimentary assembler called Initial Orders.
  - Machine instructions consisted of a single alphabet letter operation code, a decimal address, and a terminating letter, which caused one of 12 constants preset by the programmer to be added to the address at assembly time.

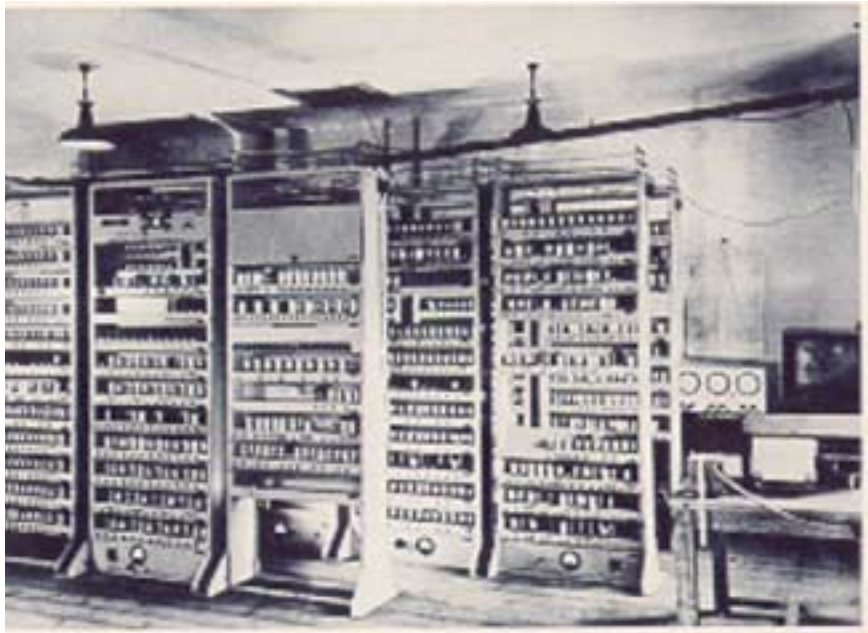
Example: Compute $C_1*(C_2+C_3*C_4)$, where C_i is value in location i.

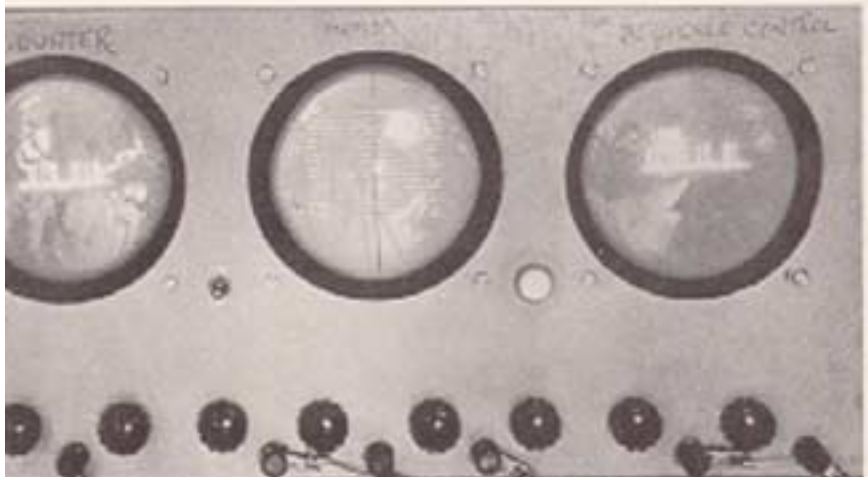
```
A   2   F      accumulator=C2
H   3   F      multiplier=C3
V   4   F      accumulator+=multiplier*C4
T   5   F      C5=accumulator
H   1   F      multiplier=C1
V   5   F      accumulator+=multiplier*C5
```

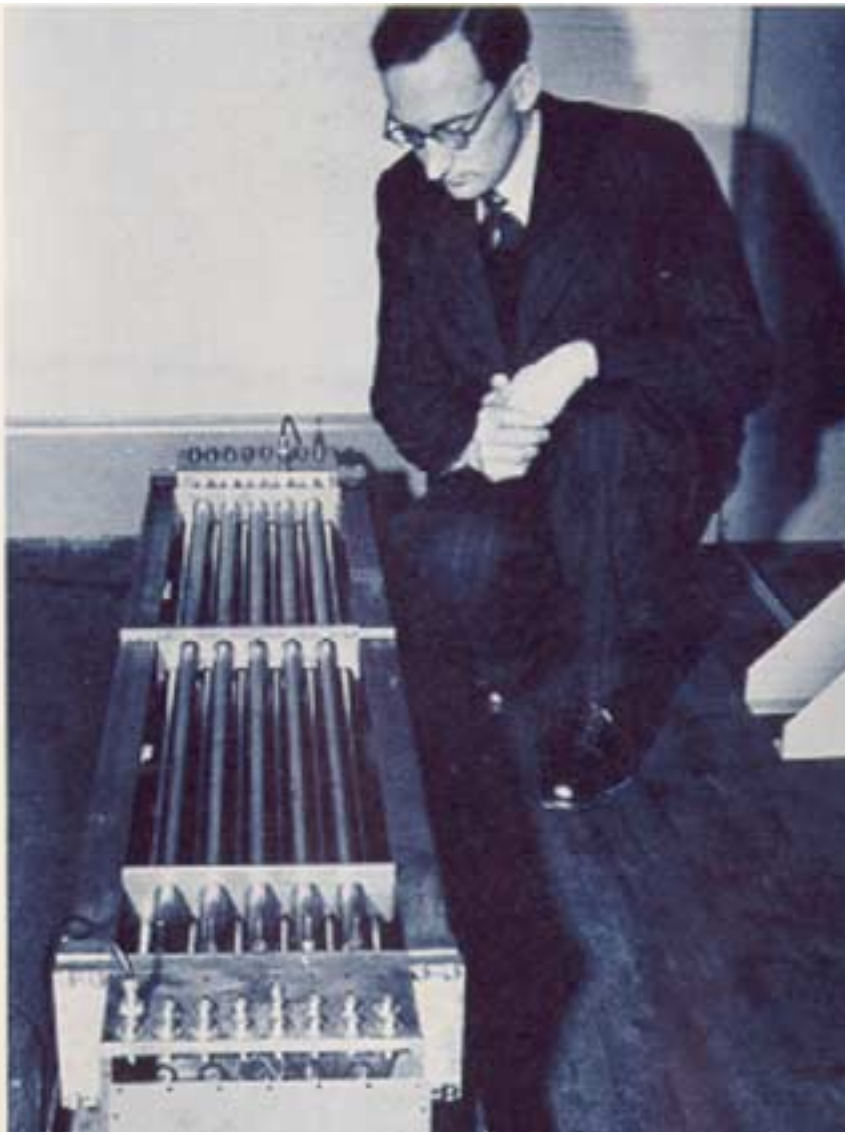
- The F is used to indicate the end of an order.
- The assembler mapped letters onto operation codes and decimal addresses onto binary form.

- The first real assembler was SOAP (Symbolic Optimizer and Assembly program) on the IBM 650 in the mid 1950s.

- The Symbolic Assembly Program (SAP) for the IBM 704 which set the external form of an assembly language that was to be a model for all its successors and which persists almost unchanged to the present day.







Fortran I

- The cost of developing machine language programs (exacerbated as machines grew in power and programs became longer) led to the development of high-level languages.
- The first commercial optimizing compiler was developed by and IBM team led by John Backus for the FORTRAN (FORmula TRANslator) between 1954 and 1957.
- Fortran was designed for numerical computing and is still today one of the most popular languages in this problem domain.

"It was our belief that if FORTRAN, during its first months, were to translate any reasonable "scientific" source program into an object program only half as fast as its hand coded counterpart, then acceptance of our system would be in serious danger. This belief caused us to regard the design of the translator as the real challenge, not the simple task of designing the language."...

"To this day I believe that our emphasis on object program efficiency rather than on language design was basically correct. I believe that has we failed to produce efficient programs, the widespread use of language like FORTRAN would have been seriously delayed.

John Backus
FORTRAN I, II, and III
Annals of the History of Computing
Vol. 1, No 1, July 1979

“Like most of the early hardware and software systems, Fortran was late in delivery, and didn’t really work when it was delivered. At first people thought it would never be done. Then when it was in field test, with many bugs, and with some of the most important parts unfinished, many thought it would never work. It gradually got to the point where a program in Fortran had a reasonable expectancy of compiling all the way through and maybe even running. This gradual change of status from an experiment to a working system was true of most compilers. It is stressed here in the case of Fortran only because Fortran is now almost taken for granted, as it were built into the computer hardware.”

Saul Rosen
Programming Languages and Systems
McGraw Hill 1967

Algol

- Defined by an international committee in the late 1950s. The ACM selected a US representative (Alan Perlis) and there were many european representatives.
- "...they proposed that English language key words, like begin, end, for, do, be used as world-wide standard. Of course this is something the American committee would never had proposed, but it seemed quite reasonable to go along with the Europeans in this matter" (Saul Rosen)
- It was intended to be a computer independent problem-oriented language.

(Fortran was designed as a high-level language for the IBM 704 and the early versions of the language reflected the hardware constrains of the 704).

- Algol was never commercially popular, but is historically important because of the programming ideas it introduced. These include:
  - Explicit declaration of variable types
  - Block structure (the blocks could be nested and explicit declarations of variables within the blocks developed the idea of local variables).
  - Recursion

- These features can be found in many languages today including Pascal, Ada, Modula-2, C, C++, and Java.

Imperative Languages

- Fortran and Algol are examples of imperative languages.
- The most popular programming languages are, and have always been, imperative languages. Examples include: Cobol, PL/1, C, C++, Modula-2, Pascal, MATLAB, Simula, and Java.
- Perhaps, the most important characteristic differentiating imperative languages is that they enable the direct control of amount of memory used by the program and the value of each memory location.

- Imperative languages have evolved over the years incorporating important new features such as:
  - Vector operators (introduced by APL [A Programming Language] and now found in Fortran 95).
  - Data abstraction, objects. Smalltalk, C++, Java.
  - Concurrency. Synchronization, locking, and parallel trheads and loops.

Functional and Logic Programming Languages

- There are two families of declarative languages. Functional languages use the language of mathematical functions to provide their meaning. Programs are written in the form of functions, each of which returns a value. The meaning of a function call is defined in terms of the value returned. So the meaning of a function can be seen as a declaration of its returned value.

- Perhaps the most popular and influential functional language is Lisp, developed by John McCarthy and his associates at M.I.T. during the late 1950s and early 1960s.
- Since then, many functional languages have been developed. Examples include: Scheme (a dialect of Lisp developed by Gerald Sussman and Guy Steele also at MIT), ML (a very popular language in academic circles today), SISAL (developed at Lawrence Livermore National Laboratory for scientific computing by Jim McGraw and his coworkers).

- Logic languages, as their name suggests, rely on logic to provide the framework for their meaning. Logic itself was invented as an aid to human thought, allowing knowledge to be set out and reasoned about. It would therefore seem natural to attempt to use it in programming computers. In practice, first order predicate calculus is used, or at least a subset of it. It is not necessary to understand predicate calculus before learning a logic programming language.
- Prolog grew (early 1970s) out of work on natural language processing done by Alain Colmerauer in Montréal and later in Marseille, and separate work on the use of

logic for programming by Robert Kowalski at Imperial College, London.

- Logic-based languages had a prehistory with U.S. attempts such as Micro-Planner (in which Terry Winograd built the pioneering blocks world natural language processing system, SHRDLU) and Conniver. Both attempts failed to replace LISP as an artificial intelligence programming language because they were extremely inefficient. The fact that Prolog overcame the problems that bedevilled Micro-Planner and Conniver is due to the work by David H D Warren and others at the University of Edinburgh on the efficient implementation of Prolog. Warren's

name has passed into logic programming in the name given to Prolog's commonly used implementation technique, the Warren Abstract Machine.