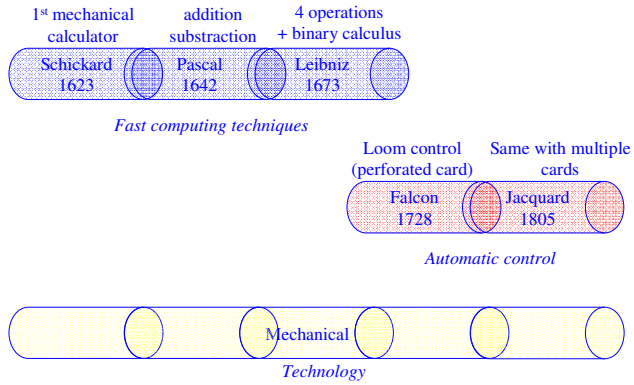
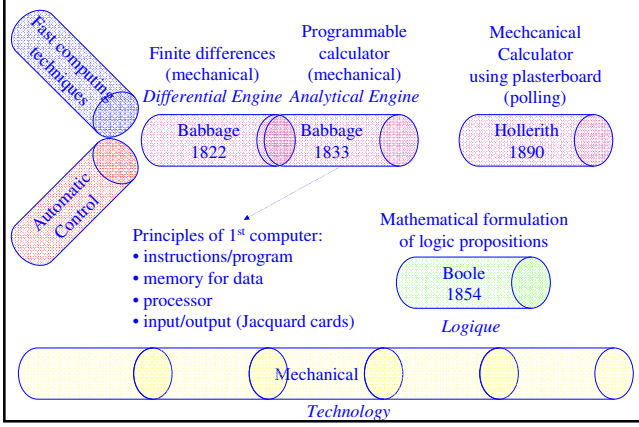
Computers Evolution - History

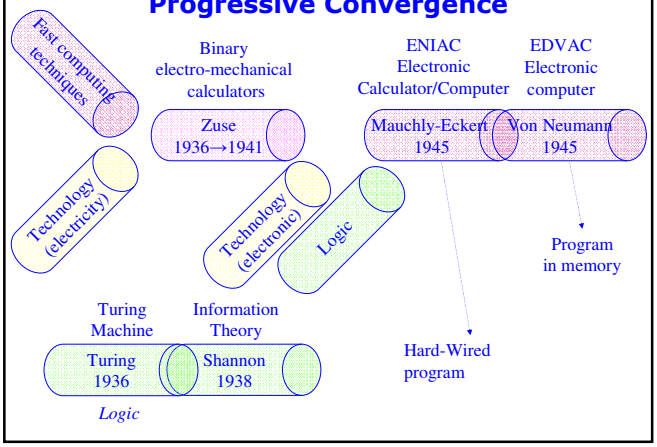
History
Progressive Convergence



Progressive Convergence

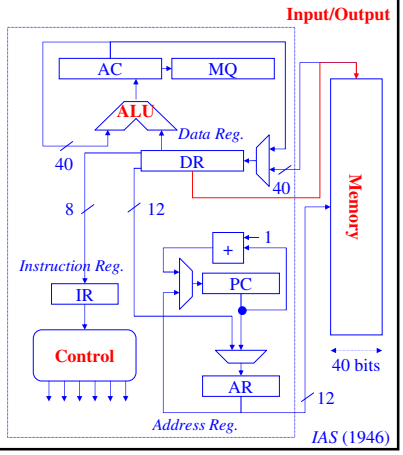


History
Progressive Convergence



Von Neumann Model

- Program (and data) in memory
- Program-guided execution
- « Universal » machine
- Branching instructions
- (Vacuum tubes)



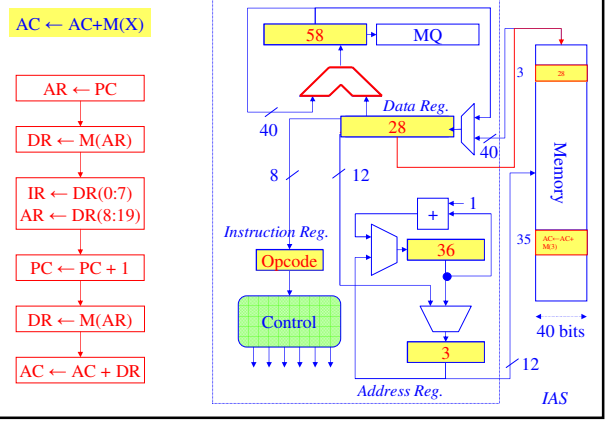
IAS (1946)

IAS – Instruction Set

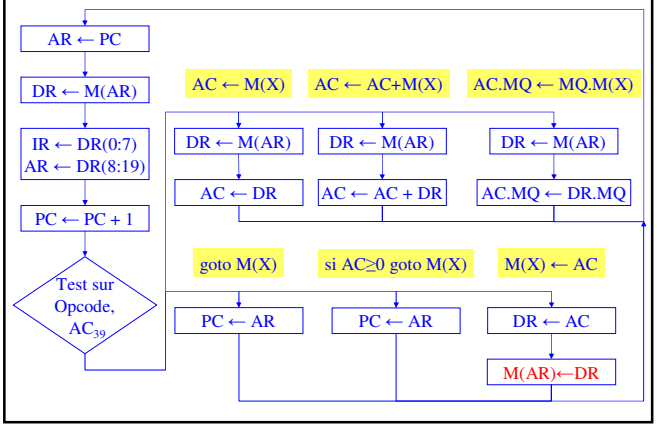
- 40-bit word
- 20-bit instructions: 8-bit operation code (*opcode*), 12-bit address
- Maximum memory size: $2^{12} \times 40$ = 163840 bits.

- `AC ← M(X)`
- `M(X) ← AC`
- `goto M(X)`
- `if AC ≥ 0 goto M(X)`
- `AC ← AC +/− M(X)`
- `AC.MQ ← MQ ×/÷ M(X)`
- `AC ← AC <</>> 1`
- `M(X,8:19) ← AC(0:11)`

IAS – Example of Operation



IAS – Control



IAS - Programming

```
for (i=0; i < 100; i++) {
   a[i] = a[i] + 5
}
```

Initializations:

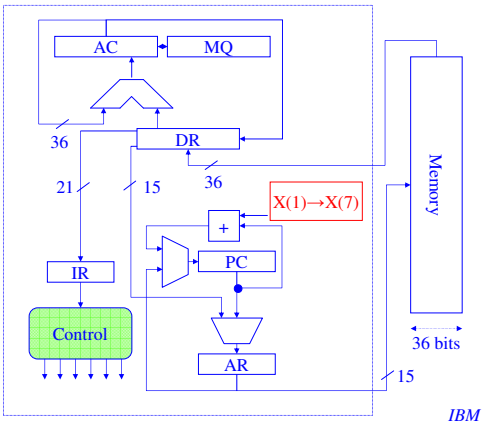
- M(20) ← 5
- M(21) ← 99 (i)
- M(22) ← 1
- M(23) ← 30
- M(30)… M(129) ← a[0]… a[99]

M(0):	AC ← M(129)	\\ AC ← a[i]
M(1):	AC ← AC + M(20)	\\ AC ← AC + 5
M(2):	M(129) ← AC	\\ a[i] ← AC
M(3):	AC ← M(21)	\\ AC ← i
M(4):	AC ← AC - M(22)	\\ AC ← AC - 1
M(5):	M(21) ← AC	\\ i ← AC
M(6):	if AC ≥ 0 goto 8	\\ if i ≥ 0 PC ← 8
M(7):		\\ end
M(8):	AC ← AC + M(23)	\\ AC ← i + @A (=@a[i])
M(9):	M(0,8:19) ← AC(0:11)	\\ Modify @a[i] in M(0)
M(10):	M(2,8:19) ← AC(0:11)	\\ Modify @a[i] in M(2)
M(11):	goto 0	\\ PC ← 0

Summary

- Main purposes: scientific computing and management
- Limitations of first computers:
  - Hard to program:
    - Simple control constructs (loops) hard to write
    - Small memory (limited array size)
  - Technology (vacuum tubes)
- 10 to 15 years later (≈1960):
  - Technology = transistors (boards)
  - Easier programming thanks to architecture and ISA improvements
  - High-Level languages and compilers
  - Input/Output management
  - Example: IBM 7094

IBM 7094



IBM 7094

IBM 7094 – ISA

- 36-bit word
- 36-bit instructions: 21-bit opcode, 15-bit address:
  - More instructions
  - Maximum memory size: $2^{15} \times 36 = 1179648$ bits.
- Loop support:
  - `TXI address, n, inc` :
    - `X(n) ← X(n) + inc`
    - `PC ← address`
    - `≈ i = i + 1 and goto label`
  - `TXL address, n, val` :
    - `if X(n) ≤ val, PC ← address`
    - `else PC ← PC + 1`
    - `≈ if i ≤ val goto label`
- Indexed addressing:
  - 7 index registers X(1) à X(7).
  - 3 opcode bits → # index register:
    - 000: no indexed addressing.
    - 001 à 111: # register.
  - `LDA address, n` :
    - `AC ← M(address – X(n))`
- Structured/Procedural programming:
  - `LINK TSX address, n` :
    - `X(n) ← PC`
    - `PC ← address`
  - `TRA i, n` :
    - `PC ← X(n) + i`
- Immediate addressing:
  - `AXT val, n` :
    - `X(n) ← val`

IBM 7094 - Programming

```
for (i=0; i < 100; i++) {
   a[i] = a[i] + 5
}
```

Initializations:
- M(20) ← 5
- M(30)… M(129) ← a[0]… a[99]

```
M(0):  AXT 99,1          \\ X(1) ← 99
M(1):  CLA M(20), 0      \\ AC ← 5
M(2):  ADD M(129), 1     \\ AC ← AC + M(129-X(1))
M(3):  STO M(129), 1     \\ M(129-X(1)) ← AC
M(4):  TXL 6, 1, 0       \\ if X(1) ≤ 0, PC ← 6
M(5):  TXI 1, 1, -1      \\ X(1) ← X(1) – 1, PC ← 1
M(6):  HPR               \\ end
```

High-Level Languages and Compilers

- FORTRAN (*FOR*mula *TRAN*slation).
- COBOL (*CO*mmon *B*usiness *O*riented *L*anguage).
- Compilation as efficient as assembly programming
- Consequences on architecture evolution:
  - Compilers → architect no longer worries about *ease of programming*
  - Transistors and integrated circuits → high increase of architecture performance and capacity

  ⇒ Architecture evolution now essentially determined by quest for more **performance**

```
for (i=0; i < 100; i++) {
   a[i] = a[i] + 5
}
```

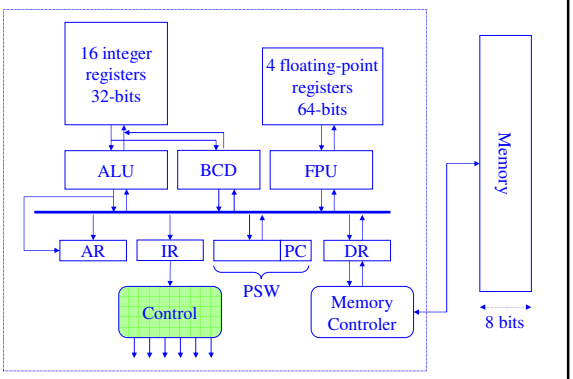
↓

```
   DO 10 I = 1, 100
      A(I) = A(I) + 5
10 CONTINUE
```

Summary

- Programming languages:
  → Widespread use of computers
- Input/Output processors → Batch mode
- 10 to 15 years later (≈1970):
  - Integrated circuits
  - Custom machines → general-purpose machines
  - Computer evolution = evolution of ISA → program binary compatibility issue
  - Growing needs → more memory
  - More complex and multi-user architectures: operating system (tasks and input/output management)
  - Example: IBM S/360.

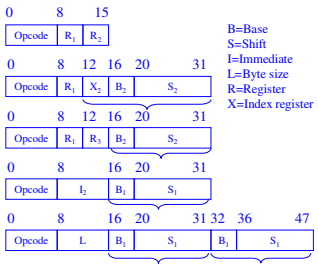
IBM S/360



IBM S/360

IBM S/360 – ISA

- All units in bytes (8 bits):
  - Character representation (8 bits)
  - All data and instruction sizes in bytes
- 32-bit word
- 16 or 32-bit integers
- Variable-Size character strings
- Variable-Size byte lists using BCD (*Binary Coded Decimal*).
- 32, 64 or 128-bit floating-point numbers
- 16 32-bit integer registers (indexing, computations)
- 4 64-bit floating-point registers
- Variable-Size instructions: 2, 4 or 6 bytes
- 5 different operand formats:

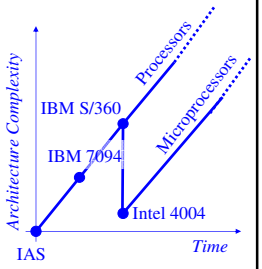


B=Base
S=Shift
I=Immediate
L=Byte size
R=Register
X=Index register

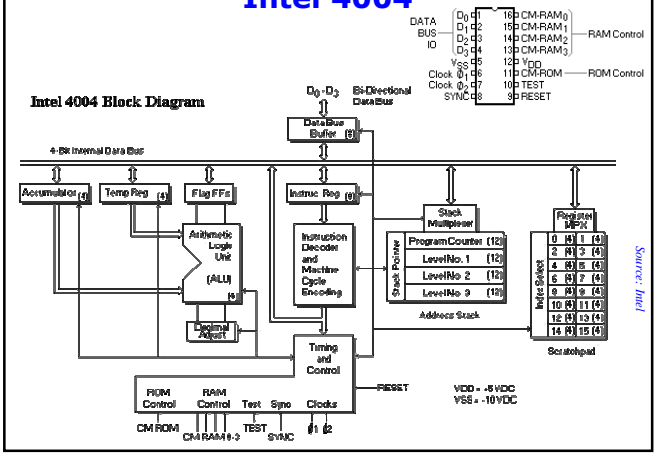
⇒ Complex ISA

Microprocessors

- Put whole processor on a single chip
- Initial goal: size, cost
- Current goal: performance, size & cost
- First microprocessor: Intel 4004
- Not the most sophisticated processor architecture

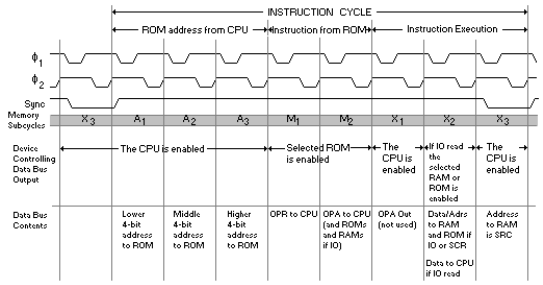


Intel 4004



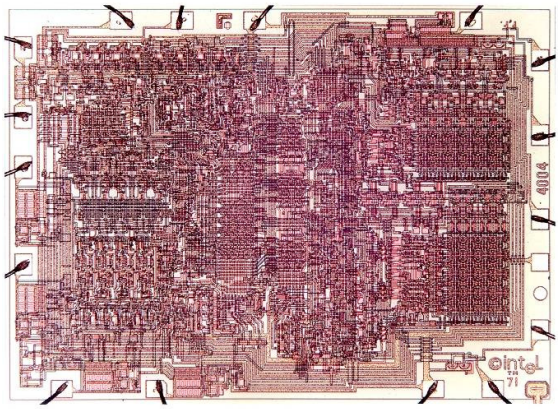
Source: Intel

Intel 4004



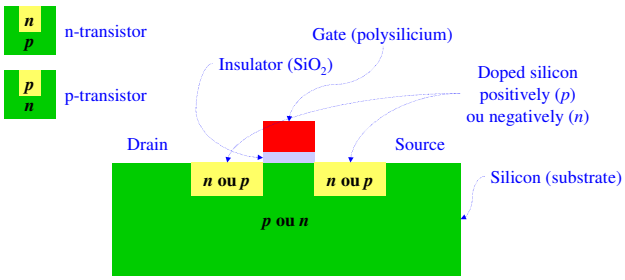
Instruction execution stages

Intel 4004



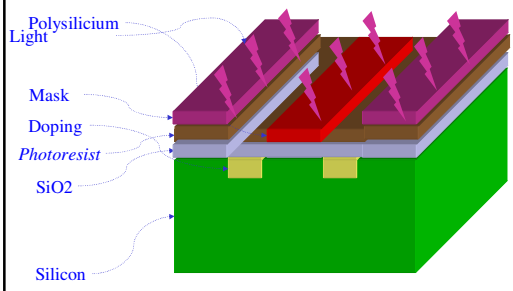
2250 transistors

Integrated Circuits
Transistors

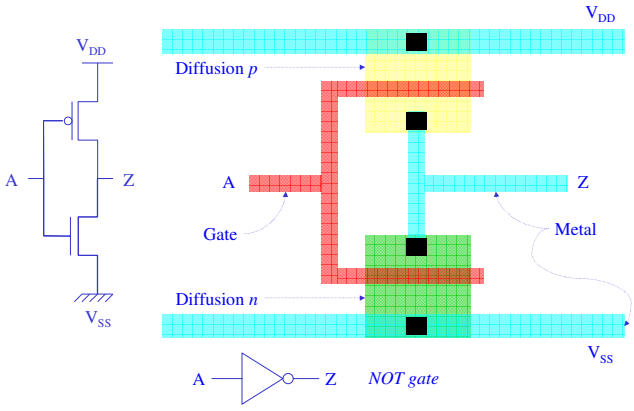


- Silicon = semiconductor
- Electrical properties modified by positive or negative doping

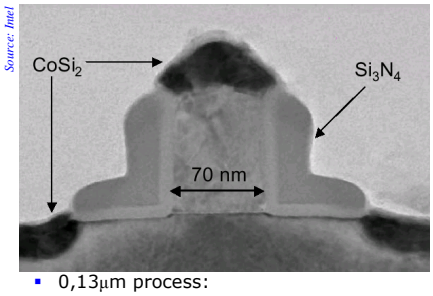
Integrated Circuits
Fabrication



Integrated Circuits
Gates

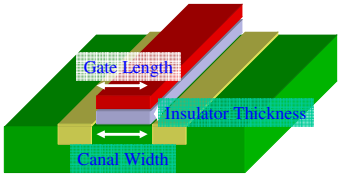


Integrated Circuits Example: 0,13µm



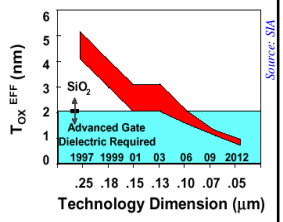
- 0,13µm process:
  - photolithography: between UV et X-ray
  - Gate length = 0,07µm
  - Insulator thickness = 1,5nm
  - Gate delay ≈ 7ps

Integrated Circuits Evolution



- Decreasing gate length:
  - ⇒ Insulator thickness must decrease
  - ⇒ Lower power consumption
  - ⇒ Shorter propagation/switching time: better performance
- Photolithography: wavelength must decrease with transistor dimension
- The gate length/insulator thickness ratio must remain roughly constant

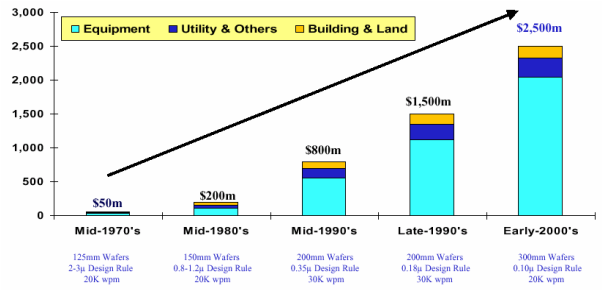
Integrated Circuits Evolution



- Moore's law: transistor size divided by two every ~ 24 months
  ⇒ Processor performance roughly doubles every 24 months
- Insulator thickness closed to limit size:
  → Other insulating materials ?
  → Other technologies ?

Année de production	2001	2004	2007	2010	2013	2016
Génération technologique : largeur de ligne dans un réseau d'interconnexions dense (nm)	130	100	65	45	32	22
Surface de cellule DRAM (surface d'un bit) (µm²)	0.14	0.049	0.03	0.012	0.0041	0.0019
Longueur physique de grille des microprocesseurs (nm)	65	37	25	18	13	9
Précision sur la longueur de grille (nm)	5.3	3.0	2.0	1.5	1.1	0.7
Épaisseur d'un isolant de grille équivalent à du SiO_2 (nm)	1.3-1.6	0.9-1.4	0.6-1.1	0.5-0.8	0.4-0.6	0.4-0.5
Tension d'alimentation (Volt)	1.2	1.0	0.7	0.6	0.5	0.4
Vitesses intrinsèques des transistors (ps, 10^{-12} seconde)	1.6	0.99	0.68	0.39	0.22	0.15

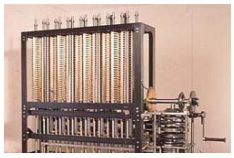
Integrated Circuits Cost

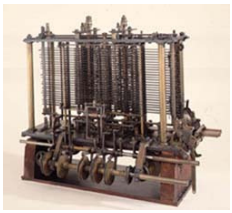


Source: Future Horizons, www.futurehorizons.com

- Fab cost

Programme pour l'*Analytical Engine*



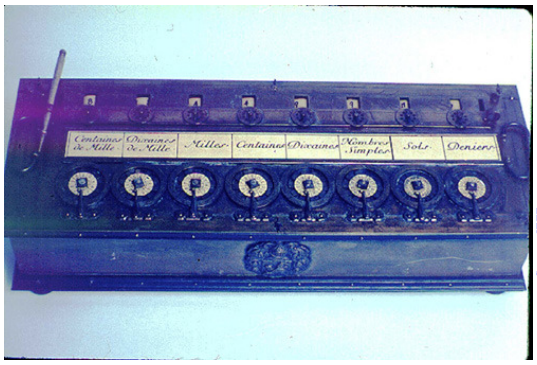


$$a_{11}x_1 + a_{12}x_2 = b_1$$
$$a_{21}x_1 + a_{22}x_2 = b_2$$

$$x_1 = \frac{a_{22}b_1 - a_{12}b_2}{a_{11}a_{22} - a_{12}a_{21}}$$
$$x_2 = \frac{a_{11}b_2 - a_{21}b_1}{a_{11}a_{22} - a_{12}a_{21}}$$

DATA		PROGRAM			
		OP.	VARIABLE CARDS		COMPUTATION
LOC.	CONTENTS	CARD	SOURCE	DEST.	
w_0 =	a_{11}	x	w_2, w_4	w_8	$w_8 \leftarrow a_{22}b_1$
w_1 =	a_{12}	x	w_1, w_5	w_9	$w_9 \leftarrow a_{12}b_2$
w_2 =	b_1	x	w_0, w_4	w_{10}	$w_{10} \leftarrow a_{11}a_{22}$
w_3 =	a_{21}	x	w_1, w_3	w_{11}	$w_{11} \leftarrow a_{12}a_{21}$
		−	w_8, w_9	w_{12}	$w_{12} \leftarrow a_{22}b_1 - a_{12}b_2$
w_4 =	a_{22}	−	w_{10}, w_{11}	w_{13}	$w_{13} \leftarrow a_{11}a_{22} - a_{12}a_{21}$
w_5 =	b_2	÷	w_{12}, w_{13}	w_{14}	$w_{14} \leftarrow w_{12} \div w_{13}$

Pascal Machine



Source: IEEE