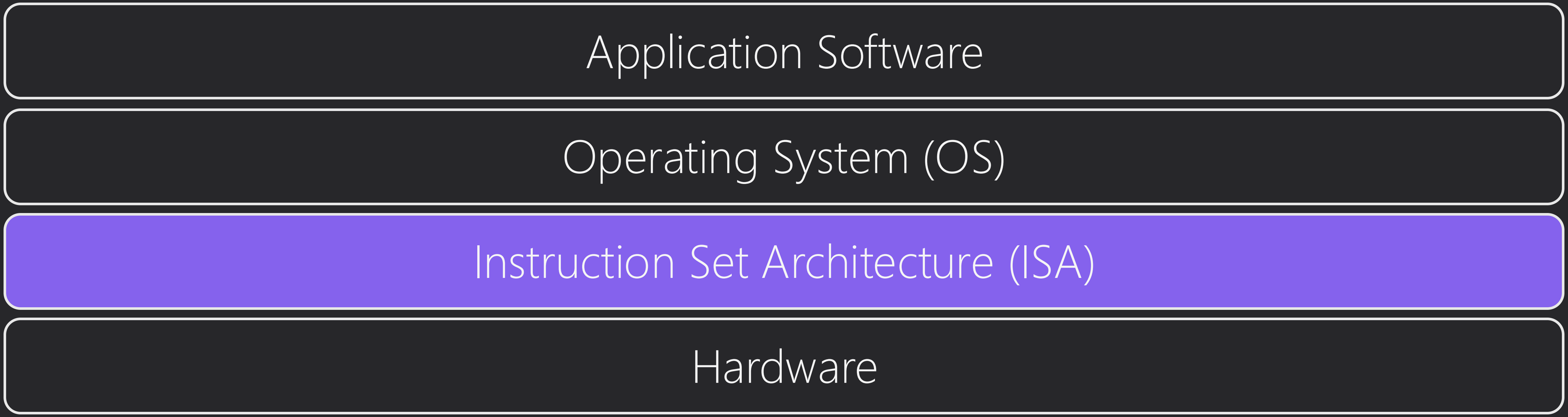
# The IBM System/360

The idea of an Instruction Set Architecture to express computation

# Managing complexity with abstraction layers



Application Software

Operating System (OS)

Instruction Set Architecture (ISA)

Hardware

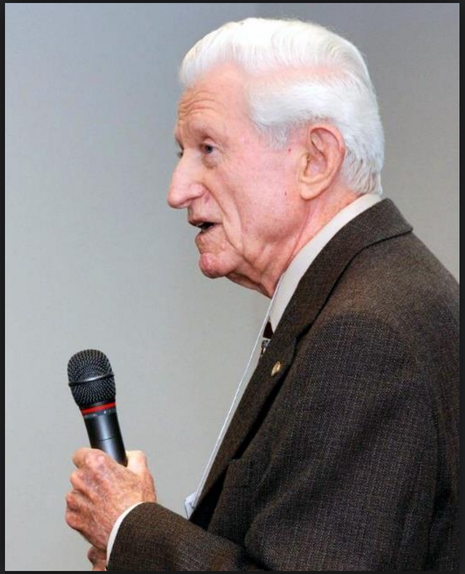
# IBM's predicament and gamble

| Model | Year(s) | Domain | Notes on instruction set |
|---|---|---|---|
| IBM 701 | 1952 | Scientific computing | • Instructions: 18-bit length<br>• Data: 18- or 36-bit sign-magnitude, fixed-point numbers<br>• Two programmable registers |
| IBM 702 | 1953-1955 | Business data processing | • Instructions: 5-character length<br>• Data: Variable-length character strings<br>• Two programmable registers (512 characters each) |
| IBM 704 | 1954 | Scientific computing | • Instructions: 36-bit length<br>• Data: Support for floating point<br>• Several programmable registers, with different lengths<br>• The high-level languages FORTRAN and LISP first developed for this machine |

- What is the problem?
- The "gamble" would cost 100s of billions of dollars in today's money
- What was the gamble?

# The gamble: Architecture of the IBM System/360

Gene Amdahl



By Perry Kivolowitz. CC BY 3.0

Gerrit Blaauw

By HandigeHarry. Public Domain

Fred Brooks

By SD&M. CC BY-SA 3.0

# The four IBM System/360 innovations

From the paper abstract:

1. An approach to storage which permits and exploits very large capacities, hierarchies of speeds, read-only storage for microprogram control, flexible storage protection, and simple program relocation.
2. An input/output system offering new degrees of concurrent operation, compatible channel operation, data rates approaching 5,000,000 characters/second, integrated design of hardware and software, a new low-cost, multiple-channel package sharing main-frame hardware, new provisions for device status information, and a standard channel interface between central processing unit and input/output devices.
3. A truly general-purpose machine organization offering new supervisory facilities, powerful logical processing operations, and a wide variety of data formats.
4. Strict upward and downward machine-language compatibility over a line of six models having a performance range factor of 50

# What’s next?

- A look at the IBM System/360 and how it established the ISA as an interface that abstracts the hardware
- A summary of important design decisions made in the IBM System/360
- A reflection on the impact of the IBM System/360

# Design objectives

> “The functions of the central processing unit (CPU) proper are specific to its application only a minor fraction of the time.”

# Inter-model compatibility

A definition of “strictly program compatible” from the paper:

> A valid program, whose logic will not depend implicitly upon time of execution and which runs upon configuration A,  will also run on configuration B if the latter includes at least:
>
> - the required storage,
> - the required I/O devices, and
> - the required optional features

# A general-purpose machine structure

Each block shows how a model might be configured.

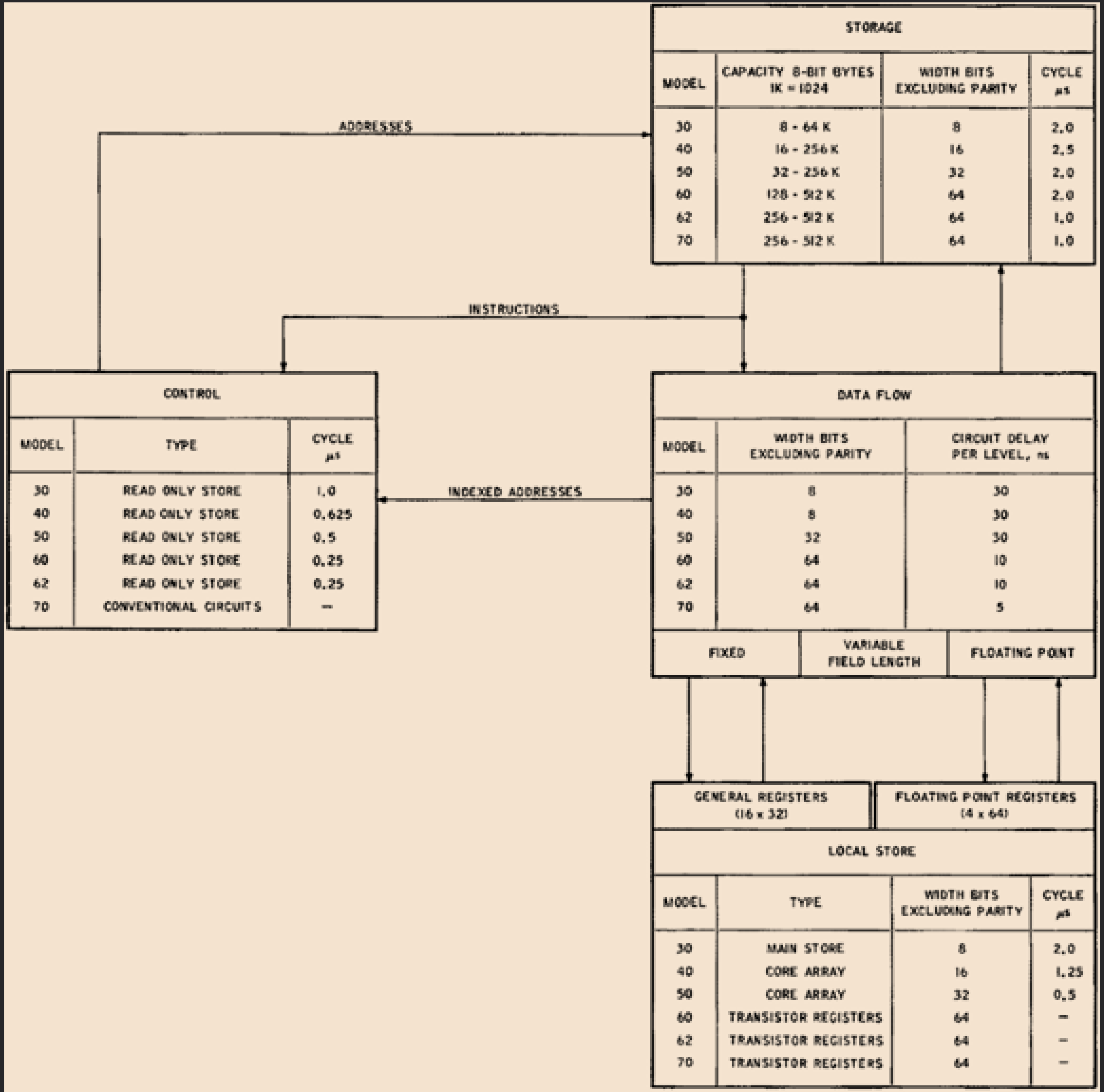
- Model 30 is smaller/slower/cheaper than the Model 70

Note the registers; *every* model has:

- Sixteen 32-bit general-purpose registers
- Four separate 64-bit floating point registers

Note the changes in width:

- Data flow width: Model 30’s 8-bit vs Model 70’s 64-bit
- Storage width: Model 30’s 8-bit vs. Model 70’s 64-bit



STORAGE

| MODEL | CAPACITY 8-BIT BYTES 1K = 1024 | WIDTH BITS EXCLUDING PARITY | CYCLE µs |
|---|---|---|---|
| 30 | 8 - 64 K | 8 | 2.0 |
| 40 | 16 - 256 K | 16 | 2.5 |
| 50 | 32 - 256 K | 32 | 2.0 |
| 60 | 128 - 512 K | 64 | 2.0 |
| 62 | 256 - 512 K | 64 | 1.0 |
| 70 | 256 - 512 K | 64 | 1.0 |

CONTROL

| MODEL | TYPE | CYCLE µs |
|---|---|---|
| 30 | READ ONLY STORE | 1.0 |
| 40 | READ ONLY STORE | 0.625 |
| 50 | READ ONLY STORE | 0.5 |
| 60 | READ ONLY STORE | 0.25 |
| 62 | READ ONLY STORE | 0.25 |
| 70 | CONVENTIONAL CIRCUITS | — |

DATA FLOW

| MODEL | WIDTH BITS EXCLUDING PARITY | CIRCUIT DELAY PER LEVEL, ns |
|---|---|---|
| 30 | 8 | 30 |
| 40 | 8 | 30 |
| 50 | 32 | 30 |
| 60 | 64 | 10 |
| 62 | 64 | 10 |
| 70 | 64 | 5 |

LOCAL STORE

| MODEL | TYPE | WIDTH BITS EXCLUDING PARITY | CYCLE µs |
|---|---|---|---|
| 30 | MAIN STORE | 8 | 2.0 |
| 40 | CORE ARRAY | 16 | 1.25 |
| 50 | CORE ARRAY | 32 | 0.5 |
| 60 | TRANSISTOR REGISTERS | 64 | — |
| 62 | TRANSISTOR REGISTERS | 64 | — |
| 70 | TRANSISTOR REGISTERS | 64 | — |

Amdahl, Gene M., Gerrit A. Blaauw, and Frederick P. Brooks. "Architecture of the IBM System/360." IBM Journal of Research and Development 8.2 (1964): 87-101.

- Size of a character
- Size of a float
- Memory alignment
- Register model

# Design decisions

An evaluation of trade-offs

# The size of a character

- Remember: memories are still “small” – the size of data matters
- The size of a character is the minimum addressable element
- Alphanumeric characters need 6 bits
- Decimal digits need 4 bits

- The 6-bit character
  - Wastes 2 bits on decimal digits
  - Used in IBM 702-7080 and 1401-7010 families

- The 4-/8-bit character
  - 4 bits used for decimal digits
  - 8 bits used for alphanumeric (2 bits wasted)
  - Used in the IBM 650-7074 family

# The size of a floating-point number

- The size of a floating-point number impacts the
  - time to perform operations like addition and multiplication (larger is longer)
  - precision of represented numbers (larger is more precise)

- 48-bit floating point
  - No option between “low” and “high” precision

- 32-/64-bit floating point
  - The client decides on the speed/space versus tradition trade-off
  - 32-bit is “single precision”
  - 64-bit is “double precision”

# The alignment of data in memory

- Different models have different widths, so different preferences for alignment
  - Model 30 would prefer 8-bit alignment
  - Model 70 would prefer 64-bit alignment

- Recall that memory is byte addressable (size of a character is 8 bits)

- The adopted rule: Each fixed field must begin at a multiple of its field length
  - e.g., all memory addresses of doubles (64 bits = 8 bytes) are found at multiples of 8

# Pushdown stack versus addressed registers

## Pushdown stack

- Operands to an operation are implicit

e.g, Z = X + Y in a stack architecture:

```
push X    # stack[top++] = mem[X]
push Y    # stack[top++] = mem[Y]
add       # stack[top++] = stack[--top] + stack[--top]
pop Z     # mem[Z] = stack[--top]
```

## Addressed registers

- Operands to an operation are explicit

e.g., Z = X + Y in a System/360 model:

```
Load R1, X                     # R1 = mem[X]
Add R2, R1, Y                  # R2 = R1 + mem[Y]
Store R2, C                    # mem[C] = R2
```

# Other design decisions

- Addresses are in binary, not decimal

- Several design decisions that allowed for an “operating system” to work
  - Asynchronous operation of I/O (via interrupts)
  - Supervisor calls

# Conclusion

What are the implications?

# The design decisions “stuck”

## The “small” decisions

- The 8-bit byte
- Byte-addressable memory
- 32-bit words, 32-bit floats, 64-bit doubles
- Addressed general-purpose registers

## The “big” decisions (gambles)

- Separate the instruction set architecture from its hardware implementation
- Ensure binary compatibility across models with different specs
- The Basic Operating System BOS/360 released in 1965 (a year after the System/360)
    - DOS/360, its successor, was the most widely used OS in the world

# The IBM Telum II Processor

Introduced at Hot Chips 2024:

- Eight 5.5 GHz cores
- Ten 36 MB L2 caches
- An I/O accelerator
- An AI accelerator

- Backwards compatible with the IBM System/360!