ETH zürich

Architecture of the IBM System/360

IBM Journal of Research and Development, 1964

Gene M. Amdahl Gerrit A. Blaauw Frederick P. Brooks Jr.

Outline

- Background
- Problems and Goals
- Key Ideas
- Design Choices
- Conclusion
- Strengths
- Weaknesses
- Thoughts and Ideas

Outline

- **Background**
- Problems and Goals
- Key Ideas
- Design Choices
- Conclusion
- Strengths
- Weaknesses
- Thoughts and Ideas

Background

Computers in the 60s:

- large mainframe computer systems
- accessed by users via terminals
- mainly used for commercial or scientific applications
- programs only ran on the system they were written for

Outline

- Background
- **Problems and Goals**
- Key Ideas
- Design Choices
- Conclusion
- Strengths
- Weaknesses
- Thoughts and Ideas

Problems

- Scientific and Business applications have different requirements
  - e.g. use of floating-point vs decimal arithmetic
  - Systems were designed and optimized for one type of application
- Computers have to keep up with the advances in storage technology and the need of large-capacity storage
  - e.g. development of magnetic recording on tapes, drums and disks
- **expensive** to upgrade
  - programs have to be rewritten for the new system

Goals

Goal: Create a system that

- allows efficient logical data processing for different applications
  - e.g. scientific, business, communication, real-time
- supports a variety of data formats
- facilitates the development of software
- offers a 50-fold performance range
- allows for cheaper upgrades
- exploits very large storage capacities and hierarchies of speed
- permits flexible storage protection and simple program relocation
- features a powerful input/output system that offers high data rates, new degrees of concurrent operation, new provisions for device status information, and much more

General-purpose utility

Compatibility across system family

Improved storage system

Powerful I/O system

Outline

- Background
- Problems and Goals
- **Key Ideas**
- Design Choices
- Conclusion
- Strengths
- Weaknesses
- Thoughts and Ideas

Key Ideas

- Make design choices that work well for small and large models, as well as for all types of applications
- Make tradeoffs to accomodate for design objectives
- Have a relative independence of logical structure and physical realization

Outline

- Background
- Problems and Goals
- Key Ideas
- **Design Choices**
- Conclusion
- Strengths
- Weaknesses
- Thoughts and Ideas

Design Choices

- Many different, sometimes small, design decisions (not all treated today)
- Design decisions often depend on each other
  - choices weren't made individually
- Many decisions were influenced by several design objectives
  - e.g. by problems arising from small and large machines
- For most decisions, the paper discusses
  - some possibilities
  - advantages/disadvantages of these possibilities
  - the reason for the choice

General-purpose

Compatibility

Storage

I/O system

Design Choices

- Data Format
  - Character size
  - Floating-point format
  - Boundary alignment

- Instruction set
  - Registers
  - Storage Addressing

- Input/Output System
  - Channel instructions
  - Standard interface & Implementation

Design Choices

- **Data Format**
  - **Character size**
  - **Floating-point format**
  - **Boundary alignment**

=> Decisions about how to represent data in memory

=> Primarily influenced by objectives for **compatibility** and **general-purpose utility**

- Instruction set
  - Registers
  - Storage Addressing
- Input/Output System
  - Channel instructions
  - Standard interface & Implementation

Data Format - Character size

Compatibility

- How many bits to represent digits or characters?
  - 10 Decimal digits => min. **4** bits for representation
  - 52 alphabetic characters => min. **6** bits for representation
- Different approaches considered
  - **4-bit decimals/6-bit characters**
    - no bits wasted
    - 6 bits for alphabetic characters = little room to extend character set
    - high engineering complexity
  - **6-bit for decimals and characters**
    - no expandability for the character set
    - 2 bits wasted for representing decimals
  - **4-bit decimals/8-bit characters**
    - no bits wasted for decimals
    - 8 bits allow for more characters in the future
    - **=> 8-bit "byte"**

Data Format - Floating point format

Compatibility

General-purpose

- **32-/64-bit vs 48-bit floating point word length**
- Longer floats
  - allow for higher precision
  - have higher computational costs
  - need more storage
- Precision of 48 bits is often not needed
- Decision:
  - offer support for 32-bit and 64-bit floats
  - let the programmer decide which one to use

Data Format - Boundary alignment

Compatibility

- Memory width varies between small and large models (8 - 64 bit)
- Boundary alignment
  - data has to aligned on proper boundaries in memory
  - guarantees efficient machine operation when a program written for one model is run on another model => **compatibility**
  - general rule:
    - fixed length data starts at a multiple of its field length
    - variable length data can start an any address
  - invalidities are hardware-detected and cause interrupts

Design Choices

- Data Format
  - Character size
  - Floating-point format
  - Boundary alignment

- **Instruction set**
  - **Registers**
  - **Storage Addressing**

=> Primarily influenced by objectives for **general-purpose utility**, **storage management** and **compatibility**

- Input/Output System
  - Channel instructions
  - Standard interface & Implementation

Instruction set - Registers

General-purpose

Compatibility

- The architecture features
  - 16 32-bit general-purpose registers
  - 4 64-bit floating-point registers
- each model has an appropriate mechanization of the same logical design (thus no problem of **compatibility**). Depending on model, registers are physically realized in
  - core storage
  - local high-speed storage
  - transistors
- The 16 general-purpose registers can be used as
  - index registers
  - relocation registers
  - accumulators for fixed-point arithmetic
  - registers for logical operations

Instruction set - Storage addressing

Compatibility

Storage

- Requirements:
  - **large machines** need to be able to address **a lot of memory**
  - **small machines** need small addresses to save space and instruction fetch time
- 24-bit byte-addressable memory space
  - allows for **16 megabytes of memory**
- base-register approach
  - 4 bits to specify base-register, containing a 24-bit address
  - 12 bits to specify displacement
    - allows relative addressing of up to 4095 bytes beyond base address
  - => 16 bits for a memory address
  - allows for easy **program relocation**

Design Choices

- Data Format
  - Character size
  - Floating-point format
  - Boundary alignment

- Instruction set
  - Registers
  - Storage Addressing

- **Input/Output System**
  - **Channel instructions**
  - **Standard interface & Implementation**

=> concerns **I/O system** and **compatibility** objectives

I/O System - Channel Instructions

- **Compatibility** issue:
  - **small machines** use CPU hardware for I/O functions
  - **large machines** use independent physical channels
- Solution: **distinction between logical and physical structures**
  - logical design considers the channel as an independently operating entity
- Idea of **channel instructions**:
  - CPU specifies the beginning of a channel program
  - channel instructions specify storage blocks to be read or written, unit operations, etc.
  - when channel program ends, channel and device status information are available.
- Idea of **command chaining:**
  - successive channel instructions
  - devices can be reinstructed faster
  - leads to **higher effective speed**

I/O System - Standard Interface & Implementation

I/O system

Compatibility

- Channel presents a standard interface to the device control unit
  - passes data, and also control and status information
- On small models, the flow of data and control information is time-shared between the CPU and the channel function
  - when I/O device sends data, the CPU is seized, dumped, used and restored
  - same function as with separate hardware
- Channel as conceptual entity
  - no cost for large number of channels
  - multiplex channel embodies up to 256 conceptual channels

Outline

- Background
- Problems and Goals
- Key Ideas
- Design Choices
- **Conclusion**
- Strengths
- Weaknesses
- Thoughts and Ideas

Conclusion

- The **goal** of the System/360 is to
  - serve as a **general-purpose** machine
  - allow for up- and downward **compatibility**
  - have new approaches to **large-capacity storage**
  - feature a powerful **I/O system**
- The **key contributions** to accomplish these goals were
  - independence of logical and physical structure
  - selection of formats, instructions, register assignments, etc. in a way that they were suitable for **different applications** and **levels of performance**

Outline

- Background
- Problems and Goals
- Key Ideas
- Design Choices
- Conclusion
- **Strengths**
- Weaknesses
- Thoughts and Ideas

Strengths

- Great **technical concepts**
  - a combination of many ideas aiming to accomplish a set of multiple objectives
- Good discussion of **alternatives**
  - makes the reader aware of other possibilities
  - shows that other implementations were considered
- Decisions and reasons that are mostly easy to understand

Outline

- Background
- Problems and Goals
- Key Ideas
- Design Choices
- Conclusion
- Strengths
- **Weaknesses**
- Thoughts and Ideas

Weaknesses

- Lack of performance evaluation
  - Performance numbers would have helped justifying some decisions
  - e.g. could have given some numbers instead of writing “speed is noticeably enhanced”
- Not always clearly written which choice they made
  - 2 paragraphes where the sentence that states the choice was either missing or extremely vague
- Uninteresting appendices
  - 4 pages of tables of operation codes, control status words, etc.

Outline

- Background
- Problems and Goals
- Key Ideas
- Design Choices
- Conclusion
- Strengths
- Weaknesses
- **Thoughts and Ideas**

Thoughts and Ideas

- Compatibility = main innovation of the System/360
- IBM System/360 was one of the first systems to have an **Instruction Set Architecture (ISA)**
  - forms, together with virtual memory, an abstraction layer between software and hardware
    - high-level code is translated to machine-language instructions
    - instructions are independent of the hardware implementation (e.g CPU)
    - programs execute correctly on systems supporting the same instruction set
  - essential for success of computers
    - allowed the existence of a software market
- General concept of ISAs basically remained unchanged since 1961

Can we improve the concept of ISA?

Thoughts and Ideas

- **Discussion**: What are the problems of the ISA and virtual memory model?
  - only program **functionality** is conveyed
  - loss of high-level semantics
    - e.g. data structures (data locality)
    - these semantics could be useful for optimizations
- Goal: have an abstraction layer between software stack and hard that keeps program semantics
- **Discussion**: How can we achieve this goal?
  - **Expressive Memory** (X-Mem)
  - **Locality Descriptor** (data locality in GPUs)
  - **Virtual Block Interface** (alternative to conventional virtual memory)

Thank you for listening!

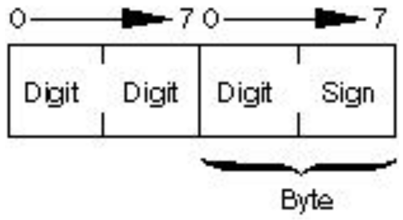
Backup Slides

Development

System/360:

- announced on April 7, 1964
- family of 6 models (9 models got added later)
- Mainframe computer systems
- 5 Billion Dollar investment (equivalent today: 40 Billion)
- More than 70.000 people hired by IBM

Data Format - Decimal Fields



- Packed-Decimal format
  - a byte of storage contains 2 decimal digits (each 4 bit)
  - low-order byte contains 1 digit and the sign
- Variable- vs fixed-length decimal fields
  - Variable length
    - higher storage efficiency and tape rate
    - performance advantage on (small) serial-by-byte machines
  - Fixed-length
    - less flexible
    - performance advantage on (large) parallel machines
- Decision: variable length
  - small machines are numerous
  - large machines are often I/O-limited => gain in tape rate compensates for the disadvantage in performance

Data Format - ASCII vs BCD code

- ASCII
  - American Standard Code for Information Interchange
  - introduced in 1963
  - 7-bit character set
- EBCDIC
  - Extended binary-coded-decimal (BCD) interchange code
  - 8-bit character set
  - easily translated from/to IBM card code
- System/360 supported both formats
  - program-selectable BCD or ASCII mode

Data Format - Floating point format

- Hexadecimal (base-16) floating point format
- Advantages compared to base-2:
  - fewer occurrences of
    - pre-shift
    - overflow
    - precision-loss post-shift
  - simpler shifting paths
- Disadvantages compared to base-2:
  - 8 times larger truncation and rounding effects
  - lower effective minimum precision
- Disadvantages can be mostly compensated by the use of 64-bit floats

Instruction set - Instruction Format

- instruction set has 5 different formats of different lengths:
  - 2 byte
    - Register-to-Register (RR)
  - 4 byte
    - Register-to-Indexed storage (RX)
    - Register-to-Storage (RS)
    - Storage and Immediate operand (SI)
  - 6 byte
    - Storage-to-Storage (SS)
- **general-purpose functionality** is ensured by having a variety of instructions for different data types