

# **Computer Architectures**

## **von Neumann Architecture**

**Memory of "words" addressed by number**

**Small set of working registers, e.g.,**

**Program counter**

**Accumulator(s)**

**Memory buffer**

**Arithmetic/logic unit**

**Performs arithmetic and other operations on data presented to it**

**Control unit**

**Fetches "instructions" from memory**

**Interprets them and executes them**

**This may involve fetching data from memory or storing data to memory or manipulating the working registers or performing operations in the arithmetic/logic unit**

**The control unit may also modify the flow of instructions depending on a testable condition**

**These are known as "branch", "jump", or "transfer" instructions**

**One key idea is that the instructions and data share the same memory**

**Since instructions are fetched and executed sequentially there is a limit on how fast such a machine can operate**

**Many modern machines use this model but use special tricks to speed up the operation**

**allow for more parallelism**

**faster arithmetic algorithms**

**main memory caching**

**bypass registers**

# **Types of Machine Instruction Architectures**

**Since data is stored in main memory by number or address there must be a mechanism for instructions to generate memory addresses for both data and instructions**

**The number of memory addresses in an instruction is a major characteristic of a machine**

**Early machines used recirculating main memories (mercury delay lines or magnetic drum) so access time to memory depended on the address and the time the request was made**

**Without special mechanisms the access time would average  $1/2$  the recirculating time**

**Some systematic references might take longer, e.g., referencing the same location repeatedly**

# **Program Counter (PC)**

**Also known as the instruction counter (IC)**

**One simple and popular method for determining the address of the next instruction is to use an incrementing counter**

**Instructions are fetched from consecutive locations unless an instruction modifies the program counter**

**If a computer has a program counter then, generally, all addresses in an instruction refer to data and not instructions (except for branching or conditional branching instructions)**

**An alternative is to have each instruction contain the address of the next instruction to be executed**

**Every instruction in such a machine is also a branch instruction**

**Hybrid machines also exist that use a combination of techniques for determining the next instruction**

# Characterizing Instruction Architecture Types

The number of addresses in a machine instruction characterizes the machine

There are "one address" machines, "two address" machines etc. If one of the addresses is used to specify the address of the next instruction this is generally specified as "+1"

There are "1+1 address" machines ... "3+1" address machines etc.

# **One Address Machines**

**The simplest addressing scheme is to have one address per instruction**

**The operation is as follows**

**An instruction is fetched from the location specified in the program counter**

**The instruction is executed, probably resulting in accessing the data at the location specified by the address in the instruction**

**... The process repeats, alternately accessing instructions and data**

**There must also be temporary accumulator register(s) for holding temporary results**

**These registers are implicitly referenced by the instruction**

**Branches and conditional branches modify the program counter appropriately (possibly by copying the address portion of the instruction into the program counter)**

**This is probably the most popular architecture**

# **1+1 Address Machines**

**One of the problems with the one address architecture in a machine with a recirculating main memory is that it is difficult to optimize the placement of instructions and data with respect to the recirculating memory**

**In a 1+1 address machine the address of the next instruction is specified in each instruction allowing complete freedom in the location of instructions**

**This allows optimally locating instructions to minimize or eliminate the latency in fetching the next instruction**

**With careful program design and location of data much of the delay associated with the recirculating main memory can be eliminated**

**Speedups of 5 times or more can be achieved**

**Programming is much more complicated**

# More Addresses per Instruction

## 2 Address Machines

It is possible to design a machine without accumulators by always returning the result to memory for each operation

A typical instruction for this machine might be

ADD A, B

where location A is added to location B and the result stored in location B

## 3 Address Machines

A typical instruction might be

ADD A, B, C

where location A is added to location B and the result stored in location C

## 2+1 or 3+1 Address Machines

For each of the above architectures it is possible to add an address specifying the location of the next instruction to be executed

# **Zero Address Machines**

**If the architecture has a stack then instructions can use the stack implicitly and no addresses need to be specified in the instruction**

**A typical instruction might be**

**ADD**

**which would pop the top two operands off of the stack, add them together, and push the sum onto the stack**

**There would also need to be instructions to push constants and memory addresses onto the stack**

# **The IBM 650**

**This machine was produced by IBM in the 1950's**

**Sold almost 2000 units**

**Cheap ~ \$500,000**

**Fits in a single room**

**User friendly - decimal and not binary**

**2000 word drum main memory (0000-1999)**

**Licensed drum technology from Remington Rand**

**1+1 Address architecture**

**50 words per track**

**Word length 10 digits plus sign**

**Two accumulators and a distributor register**

**8000 - storage entry switches**

**8001 - distributor**

**8002 - lower accumulator**

**8003 - upper accumulator**

**All data from memory flows through the distributor**

# IBM 650 Instruction Set

15 ALO - Add to lower  
65 RAL - Reset and Add to Lower  
10 AUP - Add to Upper  
60 RAU - Reset and Add to Upper  
16 SLO - Subtract from Lower  
66 RSL - Reset and Subtract from Lower  
11 SUP - Subtract from Upper  
61 RSU - Reset and Subtract from Upper  
19 MPY - Multiply  
14 DIV - Divide  
64 DVR - Divide and reset Upper  
20 STL - Store Lower Accumulator  
21 STU - Store Upper Accumulator  
22 SDA - Store Data Address of lower Accumulator  
23 SIA - Store instruction Address of lower Accumulator  
24 STD - Store Distributor  
30 SRT - Shift Right (uses D address for shift count)  
31 SRD - Shift Right and Round  
35 SLT - Shift Left  
36 SLC - Shift left and Count  
44 NZU - Branch on Non-Zero in Upper Accumulator  
45 NZE - Branch on Non-Zero Accumulator  
46 BMI - Branch on Minus Accumulator  
47 BOV - Branch on Overflow  
90-99 BD0-9 - Branch on 8 in a Distributor Position  
[Read Codes 70, 72, 73, 75, 76, 78]  
00 NOP - No Operation  
01 HLT - Halt  
69 LDD - Load Distributor  
84 TLU - table lookup  
17 AML - Add Magnitude to Lower  
67 RAM - Reset and Add Magnitude to Lower  
18 SML - Subtract Magnitude from Lower  
68 RSM - Reset and Subtract Magnitude from Lower

# **The APEXC Computer**

**"All Purpose Electronic Xray Calculator"**

**A serial computer with a magnetic drum memory**

**1+1 address architecture**

**32-bit word length**

**2's complement**

**Arithmetic unit contained two 32-bit registers**

**A - Accumulator (add and subtract and shift)**

**R - Shifting Register**

**Input/output via 5 hole teletype paper tape**

**Drum Memory**

**32 words per track and 256 tracks**

**8192 words of memory**

**One set of 512 words is permanently addressable**

**any other set of 16 tracks = 512 words can be switched into the upper part of the address space**

# Instruction set

## Instruction format

1-10 X address

11-20 Y address (generally the next instruction)

21-25 Function

26-31 C6 - a counter used in some instructions

32 vector bit

## Instructions

0 Stop

2 I(y) Input - read 5 bits from tape to top 5 bits of R

4 P(y) Punch - top 5 bits of R are punched onto tape

6 B(x)(y) Branch - goto x if A < 0 otherwise go to y

8 L(y) n Shift A and R left n places circularly

10 R(y) n Shift A and R right 64-n places and extend A sign

14 X(x)(y) Multiply track x by last 33-n digits of R to AR

16 +c(x)(y) Load x into A

18 -c(x)(y) Load -x into A

20 +(x)(y) Add x into A

22 -(x)(y) Subtract x from A

24 T(x)(y) Load x into R

26 R(x)(y) Store the first or last n bits of R in x and replace stored bits by original sign

28 A(x)(y) Store the first or last n bits of A in x and leave A unchanged

30 S(x)(y) Block head switch - switch the upper addresses to refer to the block of tracks specified in x

**There is also a vector bit in each instruction that causes the instruction to be executed 32 times, once for each location on the track specified**

# **The Early IBM Mainframe Computers**

## **IBM 701 - 1952**

**First IBM large-scale scientific computer**

**Two years from design to installation**

**The first "modern" computer**

**Electrostatic storage (later retrofitted with core memory)**

**Punched card, magnetic tape and printer IO**

## **IBM 704 - 1954 to 1960**

**Random access core storage**

**Hardware floating-point**

**3 index registers**

**First widely available "modern" machine**

## **IBM 709 - 1957 to 1960**

**Indirect addressing**

**Data channels for simultaneous IO and computing**

## **IBM 7090 - 1958 to 1969**

**Transistorized version of IBM 709**

**7.5 times faster - 2.4 microsecond core**

## **IBM 7094 - 1962 to 1969**

**7 index registers**

**Double precision floating-point operations**

**704 compatability mode**

**2 microsecond core memory**

**Basis for CTSS the Compatible Time Sharing System**

## **IBM 7040 and 7044**

**Transistorized versions of the 704 sold primarily to educational institutions**

# **IBM 704**

**A parallel 36-bit binary computer manufactured and sold in the 1950's**

**Core memory**

**12 microsecond cycle time**

**Available in units of 4096 words**

**Numbers represented as absolute value plus sign**

# Details

## Registers

**SR - Storage Register used as a buffer from core memory to the central processor**

not programmable

**AC - Accumulator was 38 bits with two extra bits of magnitude (Q and P bits) to handle overflows**

The "logical" accumulator was the P bit and bits 1 to 35 of the accumulator

**MQ - Multiplier-Quotient Register used for**

buffering for IO using the CPY instruction

the multiplier in multiplication

returns the quotient after a division

returns the least significant part of the product after multiplication

holds the least significant part of the result of a floating-point operation

**IC - Instruction Counter**

holds the address of the next instruction

**Instruction Register**

holds part of the instruction currently being executed

**Index registers**

three registers used for modifying the address of an instruction

multiple index registers could be specified in an instruction resulting in the registers being "or"ed together

**Special indicators and sense devices**

Accumulator overflow indicator

MQ overflow indicator

Divide check indicator

Tape check indicator

Trapping mode indicator

sense switches (6)

Sense lights (4)

# Instructions

## Type A instructions (5 operations)

3 bit opcode

Have two addresses

one address in the "decrement" field is used with the index registers  
the other (normal) address refers to a memory address

3 bit index field

## Type B instructions

9 bit operation field

3 bit index

15 bit address

# Type A Instructions

These instructions are useful for making loops and testing and incrementing an index register

**TXI Y** - Transfer with Index Incremented

**TXH Y** - Transfer on Index High

**TXL Y** - Transfer on Index Low or Equal

**TIX Y** - Transfer on Index

**TNX Y** - Transfer on No Index

Note that the decrement part of the instruction is what was referred to in lisp

**CDR** - Contents of Decrement register

**CAR** - Contents of Address Register

# Type B Instructions

These were standard single address instructions

There was a full set of integer, floating-point, and logical instructions

Here are the instructions

## Fixed-Point Arithmetic Operations

CLA Y - Clear and Add  
ADD Y - Add  
ADM Y - Add Magnitude  
CLS Y - Clear and Subtract  
SUB Y - Subtract  
SBM Y - Subtract Magnitude  
MPY Y - Multiply  
MPR Y - Multiply and Round  
RND - Round  
DVH Y - Divide or Halt  
DVP Y - Divide or Proceed  
LDQ Y - Load MQ  
STQ Y - Store MQ  
SLQ Y - Store Left-Half MQ  
STO Y - Store AC  
STZ Y - Store Zero  
STP Y - Store Prefix  
STD Y - Store Decrement  
STA Y - Store Address  
CLM - Clear Magnitude  
CHS - Change Sign  
SSP - Set Sign Plus  
SSM - Set Sign Minus

## Logical Operations

CAL Y - Clear and Add Logical Word  
ACL Y - Add and Carry Logical Word  
SLW Y - Store Logical Word  
AND Y - AND to Accumulator

**ANS Y - AND to Storage**  
**ORA Y - OR to Accumulator**  
**ORS Y - OR to Storage**  
**COM - Complement Magnitude**

### **Shifting Operations**

**ALS Y - Accumulator Left Shift**  
**ARS Y - Accumulator Right Shift**  
**LLS Y - Long Left Shift**  
**LRS Y - Long Right Shift**  
**LGL Y - Logical Left**  
**RQL Y - Rotate MQ Left**

### **Floating-Point Arithmetic Operations**

**FAD Y - Floating Add**  
**UFA Y - Unnormalized Floating Add**  
**FSB Y - Floating Subtract**  
**UFS Y - Unnormalized Floating Subtract**  
**FMP Y - Floating Multiply**  
**UFM Y - Unnormalized Floating Multiply**  
**FDH Y - Floating Divide or Halt**  
**FDP Y - Floating Divide or Proceed**  
**EFM - Enter Floating Trap Mode**  
**LFM - Leave Floating Trap Mode**

### **Control Operations**

**NOP - No Operation**  
**HPR - Halt and Proceed**  
**ETM - Enter Trapping Mode**  
**LTM - Leave Trapping Mode**  
**HTR Y - Halt and Transfer**  
**TRA Y - Transfer**  
**TZE Y - Transfer on Zero**  
**TNZ Y - Transfer on No Zero**  
**TPL Y - Transfer on Plus**  
**TMI Y - Transfer on Minus**  
**TOV Y - Transfer on Overflow**  
**TNO Y - Transfer on No Overflow**

**TQP Y - Transfer on MQ Plus**  
**TQO Y - Transfer on MQ Overflow**  
**TLQ Y - Transfer on Low MQ**  
**TSX Y - Transfer and Set Index**  
**TXI Y - Transfer with Index Incremented**  
**TXH Y - Transfer on Index High**  
**TXL Y - Transfer on Index Low or Equal**  
**TIX Y - Transfer on Index**  
**TNX Y - Transfer on No Index**  
**TTR Y - Trap Transfer**  
**PBT - P Bit Test**  
**LBT - Low Order Bit Test**  
**DCT - Divide Check Test**  
**RTT - Redundancy Tape Test**  
**CAS Y - Compare Accumulator with Storage <=>**

### **Indexing Operations**

**LXA Y - Load Index from Address**  
**LXD Y - Load Index from Decrement**  
**SXD Y - Store Index in Decrement**  
**PAX - Place Address in Index**  
**PDX - Place Decrement in Index**  
**PXD - Place Index in Decrement**

### **Input-Output Operations**

**RDS Y - Read Select**  
**WRS Y - Write Select**  
**BST Y - Backspace Tape**  
**BSF Y - Backspace File**  
**WEF Y - Write End of File**  
**REW Y - Rewind**  
**ETT Y - End of Tape Test**  
**LDA Y - Locate Drum Address**  
**CPY Y - Copy and Skip**  
**PSE Y - Plus Sense**  
**MSE Y - Minus Sense**

# Peripheral Specifications

## Magnetic Tapes

Up to 10 units

75 inches/second - 200 bits/inch - 7 tracks

BCD or Binary mode (even or odd parity)

3/4 inch record gap - 3.75 inch end-of-file gap

Tape may contain several files

Files may contain several records

## Magnetic Drum

Capacity of 8192 words per drum - 2 drums possible

Words are individually addressable

Must use very tight loop for reading consecutive words

## Card Punch

Punched in "row binary"

Can only punch 72 of 80 columns ( $36 \times 2 = 72$ )

## Card Reader

Read in "row binary"

Can only read 72 of 80 columns ( $36 \times 2 = 72$ )

Note limitation of FORTRAN on reading only columns 1-72 of card

## Printer

120 columns

48 characters

Carriage Control

A punched tape controlled vertical tabbing

## Cathode Ray Tube

7 inch tube

recorded output with a camera

x and y resolution of 1024 points

# Special Features

## Trapping mode

When trapping mode is set the address of every transfer instruction is stored in location 0000 and if an instruction actually causes a transfer of control then the next instruction is taken from location 0001 instead of the normal location

This allows testing a program by stopping the program on every successful transfer instruction

## CPY instruction

A CPY instruction is required to transfer each word between memory and an IO device that has been previously selected

This instruction uses the MQ register for buffering

CPY instructions must be executed in the time required by the IO device or the IO operation fails

The CPY instruction skips the next instruction if there is no more input to process

The drum is so fast that only one instruction may be executed between cpy instructions so a typical loop might be

```
LOOP  CPY  x, 1  
      TIX  LOOP, 1, 1
```

## Added Floating-Point Trap Feature - when enabled by a EFM instruction

A floating-point instruction that overflows or underflows

Stores the address of the instruction + 1 in location 0000

The next instruction is taken from 0010<sub>8</sub>

In addition the decrement portion of 0000 are set

bit 14 if overflow or underflow from a divide instruction

bit 15 if overflow occurs in either AC or MQ

bit 16 if overflow occurs in AC

bit 17 if overflow occurs in MQ

# Subroutines in the IBM 704

The TSX instruction would set an index register to (the 2's complement) of the address of the TSX instruction itself before transferring

A subroutine called with

TSX SUBR,4

could return with

TRA 1,4

A subroutine could also conveniently examine the locations around the TSX calling instruction so extra parameters could be passed in the locations following the TSX instruction

Of course the subroutine would have to return several instructions after the TSX instructions to skip over these parameter locations

# **Assembly Language Available for IBM 704 to ease coding**

**Forms had fields for**

**label - used to reference location of instruction**

**operation - symbolic opcode**

**address, tag, decrement - for setting fields in instruction**

# **The IBM "Stretch" Computer (IBM 7030)**

**Announced April 1960 and withdrawn in 1961.**

**Only 2 machines built**

**Major research effort by IBM to incorporate all advanced features of a computer**

**The following is the text of an IBM Data Processing Division press fact sheet distributed on September 27, 1960.**

The IBM 7030 Data Processing System is the fastest, the most powerful and versatile in the world. It is now nearing completion at IBM's laboratories in Poughkeepsie, New York. The first system, the original STRETCH, is being readied for the Los Alamos Scientific Laboratory under contract to the Atomic Energy Commission.

Custom-engineered IBM 7030 systems, based on STRETCH's technology, are being made available by IBM to industry and government under negotiated contract terms. Purchase price of representative systems is more than \$10,000,000 with monthly rental of more than \$300,000.

IBM 7030 computers feature unrivaled speed, memory capacity, input-output flexibility, and multiprogramming capability. These solid state systems are extremely fast and efficient in solving large technical problems. Their general purpose design also provides facilities for high speed, flexible handling of variable field length data and decimal arithmetic for commercial data processing.

An IBM 7030 system has ultra-fast transistors and circuit components. Data may be retrieved from magnetic core storage in only 2.18 millionths of a second. But it is the principle of simultaneous operation that gives the machine its truly great speed.

For example, six core storage units may be operated at the same time, quadrupling the effective speed. This permits a continuous flow of 2,000,000 instructions and 2,000,000 pieces of data a second.

The system has a new kind of random access magnetic disk storage. A disk unit holds ready for instant use, millions of words of alphabetic or numeric data. In only one second, more than 1,250,000 alphabetic characters can be read into or retrieved from the disk storage.

Peak efficiency in the handling of input-output devices is provided by a specialized computer within the system called the Exchange. This acts as a switching center. It routes information between the main core storage and as many as 32 channels, each of which can handle many input-output devices.

A high degree of overlap enables the main arithmetic unit to perform well over 1,000,000 logical operations per second. All sections of the central processing unit operate simultaneously.

Another major new unit enables the computer to look beyond the work in process and prepare for tasks to come. This is the Look-Ahead, a device that anticipates instruction and data requirements - - greatly boosting the effective memory speed. The Look-Ahead acts as a reservoir, lining up instructions and data a fraction of a

**second before they are needed, to provide a continuous information stream to the arithmetic and logical unit.**

**The IBM 7030 can also put aside what it is doing and turn in special tasks requiring immediate attention. For example, a lengthy aerodynamics problem could be temporarily interrupted for the quick processing of daily margin calls for stock brokerage firms - - or for trajectory computations of an earth satellite. The system can give priority to these calculations, while all other parallel functions continue without pause.**

**The 7030 machines have magnetic tape units, card readers, punches, and printers, as well as other special devices -- all compatible with equipment on other IBM computers.**

**A new set of generalized instructions developed for the IBM 7030 greatly simplifies the programming task. In addition, the system has more self-checking and self-correcting abilities than any other computer.**

**Despite its tremendous speed and versatility, the system takes no more floor space than the IBM 704 (2,000 sq. ft.).**

# Architecture

**Multiple core storage units (16384 words of 64-bit error correcting words) capable of simultaneous operation**

**Maximum memory size of 262144 words**

**Data could overlap word boundaries and be referenced in a single instruction**

**Double word accumulator**

**Sign "byte"**

**16 index registers**

**memory-mapped registers**

**program interrupt**

**memory protection (no memory mapping)**

**"Harvest" mechanism for scatter gather IO**

**instructions for decimal/binary conversion**

# Registers

- 0 - zero (always contained zero)**
- 1 - interval timer and time clock**
- 2 - interrupt address**
- 3 - memory protection boundary control register**
- 4 - maintenance bits ( read as zero)**
- 5 - Channel address**
- 6 - Other CPU**
- 7 - Left Zeros Count and All Ones Count**
- 8 - Upper Accumulator**
- 9 - Lower Accumulator**
- 10 - Accumulator Sign**
- 11 - Indicators**
- 12 - Mask**
- 13 - Remainder**
- 14 - Factor**
- 15 - Transit**
- 16 - 31 - Index Registers X0 to X15**

# Addressing

Words are addressed by a 18-bit address

Instructions are addressed by a 19-bit address

Each bit of the memory can be addressed directly with a 24-bit address

Addresses can be indexed by one of 15 index registers

Integer instructions can specify

- first bit of operand

- number of bits in operand

- bit offset in accumulator

Integer operands can overlap memory boundaries

Index registers also contained a count field for counting and a refill address for automatically reinitializing the index register

Immediate addressing mode that used the effective address itself as data

Indirect addressing allowed the effective address to be fetched from memory

- Multilevel indirect addressing possible

Some instructions had "relative addressing" that would address memory relative to the address of the instruction itself

- This enabled movable code segments

# **Error Correcting memory**

**Memory words were 72-bits in hardware and 64-bits to the programmer**

**The extra 8 bits were redundant and enabled the core memory unit to correct any single bit error and detect any double bit error**

**These error correcting bits could also be written on magnetic tape**

# **Parallel Operation**

**Although the model of computation was that of executing one instruction at a time the CPU operated many units in parallel**

**Up to 4 instructions could be in execution simultaneously**

**Conditional branches caused the machine to follow both paths if possible**

**Arithmetic unit was highly optimized with special algorithms for multiplication and division**

# **Interrupt system**

**Many system conditions caused interrupts**

**Interval timer**

**IO**

**Fault conditions**

**All indicators capable of causing an interrupt were collected in location 11 - the indicator register**

**Location 12 - the mask register - enabled or disabled the corresponding indicator register position to cause an interrupt**

# **Execute Instruction**

**This instruction executed the instruction at its effective address**

**If this was another execute instruction the process repeated**

**The execute instruction could be considered as calling a one instruction subroutine**

**There was also an execute instruction that stepped a pseudo instruction counter**

## **Fixed-Point Instructions**

**Could specify data length, bit position in memory, and accumulator offset**

**Some operations left a count of the one's in the result and the number of left zero's in a special register**

**All binary logical operations were provided**

## **Decimal Instructions**

**Could directly operate on BCD data**

**Could specify size of digit from 4 to 8 bits**

**Extra bits were filled from the sign byte of the accumulator**

# **Floating-Point Instructions**

**Operated on 64-bit floating-point number**

**Double precision also available**

**All numbers had three extra bits [TUV] that could be used for interrupting the program if a word with one of these bits was referenced**

**Exponent range was  $\pm 1024$  binary or about  $\pm 300$  decimal**

**Mantissa was 48 bits**

**Hardware square root**

**Noisy mode where one's were inserted during normalize operations to give indication of numerical stability**

# **Simultaneous Compute and IO**

**The exchange could manage several IO transactions simultaneously**

**Very extensive "scatter/gather" data referencing capabilities**

**Could read and write words from multiple areas of memory in a single operation**

**Much cleaner separation between computer and peripherals than in previous machines**