Historical Machines

CS520

Dept. of Computer Science
Univ. of New Hampshire
The IAS Computer

discussion based upon 1946 report by Burks, Goldstine and Van Neumann

Institute For Advanced Studies (IAS)

Princeton University
4096 40-bit words

three registers:

PC - program counter
AC - accumulator
MQ - multiplier-quotient
20-bit instructions packed two per word

- 8-bit opcode
- 12-bit address

1e one-address instruction

AC register is implicit operand
supported Fixed-point values

left-most bit is sign bit
redix point assumed to be before next bit

\[ \text{e.g. 0} \ldots \text{1111} \Rightarrow +.\text{1111} \ldots \text{1111} \]

so very much like a modern integer but with the redix point at the other end

in fact used 2's complement for negative values too
Floating-point was rejected as being too complicated.

For each value the programmer would maintain a scale factor --- similar to a modern exponent but not necessarily a power of 2.

I.e. Floating-point would be maintained in software.
Instructions

load
negate
add
subtract
multiply
divide
shift
store
unconditional/conditional branch
move
address modification
Address modification?

Change the address field of an instruction in memory using bits from AC.

This is how array indexing was done!

In the loop over an array, modify a load instruction in the loop every iteration to advance to the next array element.

Self-modifying code!
Routine self-modifying code lived on for many years,
e.g. CDC 6600 (mid-1960's design)

used RJ (return jump) instruction to do function call

```
100   RJ 200
```

UP 101 stored here

```
200
201
```

UP 200 ← this is the "return"
note: CDC 6600 did not have direct support for recursion

modern machines have run-time stacks for this purpose
Other CDC 6600 Fun Facts

- 60-bit words
- 1's complement integers
- Proprietary floating-point
- Multiple instructions per word
- 6-bit proprietary character set
but

it was a modern machine in many respects

internal parallelism

multiple functional units
multiple memory banks

load/store architecture to allow instruction overlap

load next operand while computing last operation