Minimal **I**nstruction **S**et **C**omputers

A Tale of Quirky Little Machines

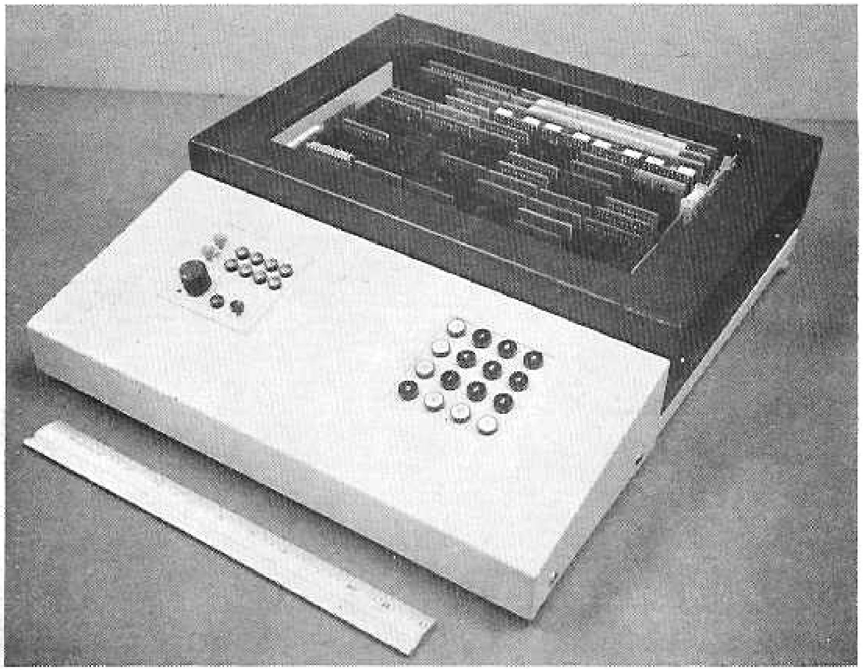
Don Roberts
SVFIG
6/23/2007

Some Useful Microprocessors

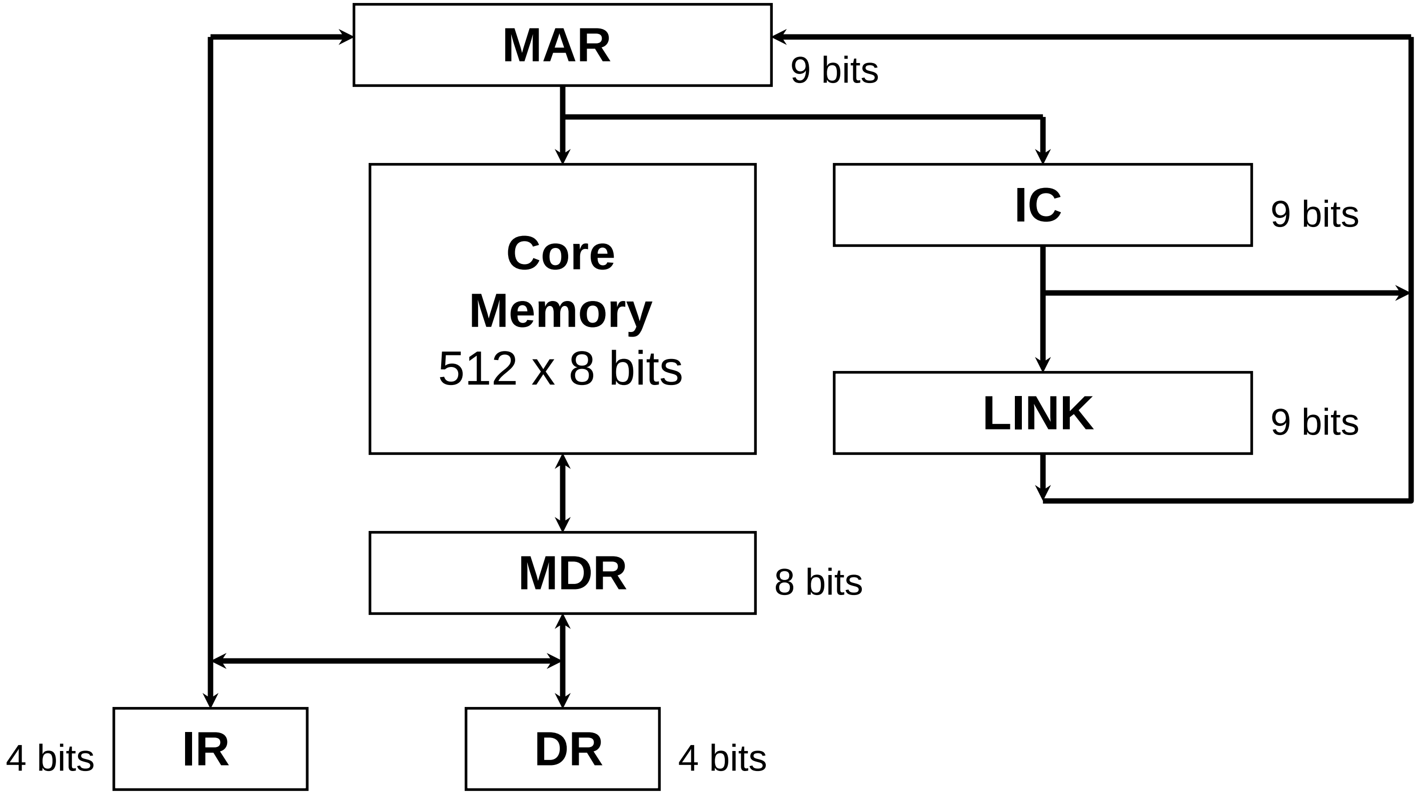
- Ting – 16 +/- 1 instructions (P8, P16, P32)
- Moore – 25 (MUP21)
- RISC-I – 39 opcodes
  - 3-operand register-to-register instructions
  - 3-stage pipeline
  - 9 ALU opcodes
    - Add, subtract, integer-inverse subtraction, AND, OR, XOR, SL-logical, SR-logical, SR-arithmetic
  - 2 Memory opcodes
    - 4 addressing modes *synthesized*
  - 4 flow-control instructions

IBM Mini-Machine

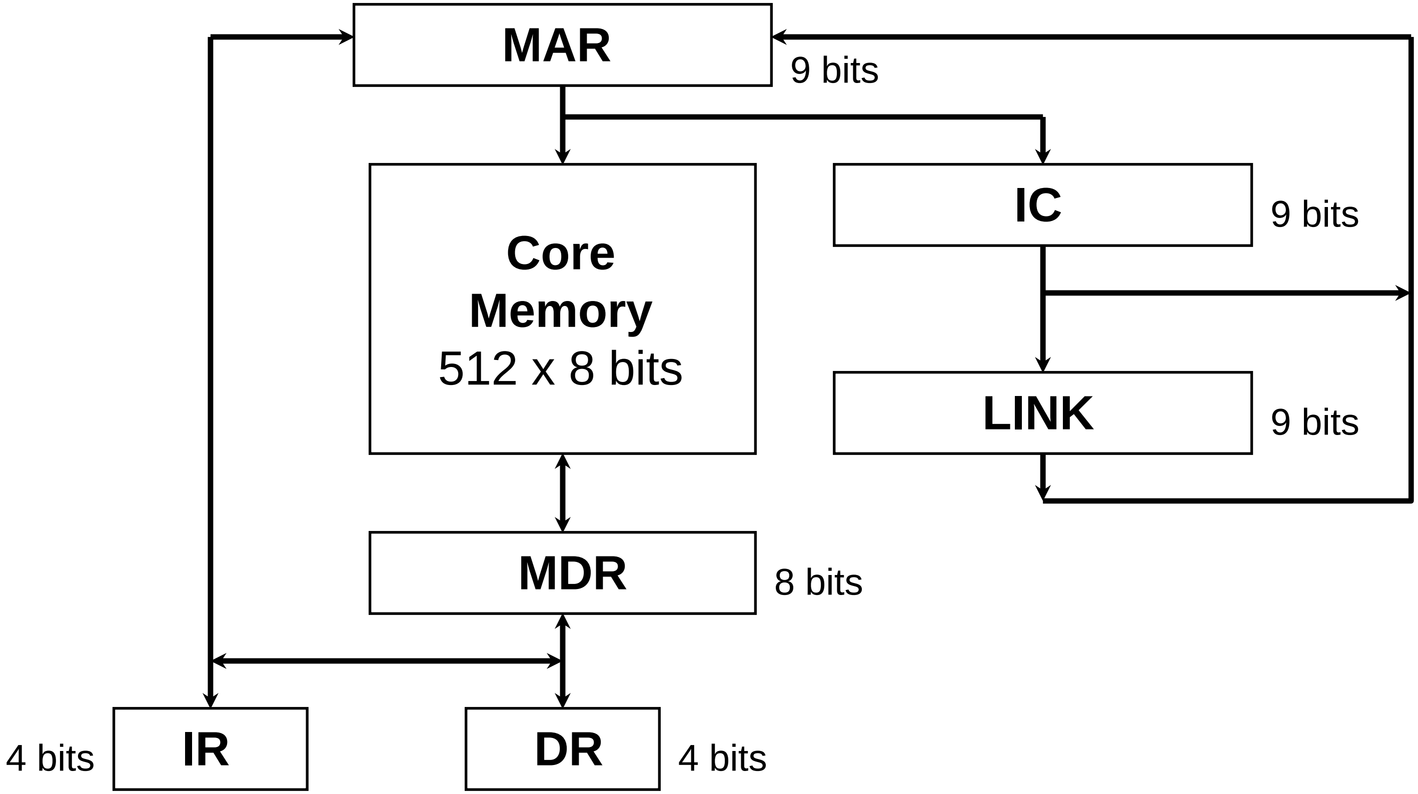
- 1968, North Carolina State
- 4-bit
- ½ KByte core (2 microsecond)
- Numeric keypad, teletype, CRT, mag-tape
- 8 instructions
  - Fetch
  - Store
  - Branch
  - Branch on zero
  - Branch on not zero
  - Branch and link
  - Return
  - Input/output



The Beastie


MAR
9 bits
IC
9 bits
Core
Memory
512 x 8 bits
LINK
9 bits
MDR
8 bits
4 bits
IR
DR
4 bits

The Beastie



Look Ma, no ALU!!

No registers either!

No ALU ?!?

- 4-bit, BCD
- All operations by table look-up & self-modifying code
  - Increment, decrement tables
  - Set, reset, flip tables
  - Shift tables
- Typically fewer than 64 bytes used up for tables
- Decimal digit add/subtract routine ~ 16 bytes

Example: Increment

Nibble	Value	Comment
X	Fetch	
X+1	Incr table address	< value to Increment
X+2	STore	
X+3	X+1	

Address	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Value	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0

Incr Table

Add-A-Digit

```
// handle carry
X1+0      Fetch
X1+1      trigger-table-start_lo
X1+2      trigger-table-start_hi
X1+3      trigger-table-offset
X1+4      BranchNot-Zero
X1+5      X4+0-lo
X1+6      X4+0-hi
X1+7      Fetch
X1+8      1              < literal
X2+0      Store
X2+1       trigger-table-start_lo
X2+2       trigger-table-start_hi
X2+3       trigger-table-offset
// do an increment
X3+0      Fetch
X3+1      incr-table-start_lo
X3+2      incr-table-start_hi
X3+3      incr-table-offset  <1st operand
X3+4      Store
X3+4      X3+3
```

```
X3+5      BranchZero
X3+6      X2+0_lo
X3+7      X2+0_hi
// count the increments
X4+0      Fetch
X4+1      decr-table-start_lo
X4+2      decr-table-start_hi
X4+3      decr-table-offset   <2nd operand
X4+4      STore
X4+5      X4+3_lo
X4+6      X4+3_hi
X4+7      BranchNotZero
X4+8      X3+0_lo
X4+9      X3+0_hi
// wrap it up
X4+10     Fetch
X4+11     X3+3_lo
X4+12     X3+3_hi
X4+13     Return
```

Some “Applications”

- Calculator
  - Keyboard input, CRT numeric display!
  - Signed add, subtract, multiply, divide
- Triangle side-angle-side
  - Graphic and alphanumeric display on CRT
  - Trig functions overlays via “mag tape”
- Vector graphics
  - Limited, but…
  - Draw with cursor, “animation”

How Low Can You Go?

- With memory mapped I/O, Mini-Machine would be 7 instructions
- Can we go lower?
- How low?

How Low Can You Go?

- With memory mapped I/O, Mini-Machine would be 7 instructions
- Can we go lower?
- How low?

Would you believe … 1

RSSB

- Reverse-Subtract, Skip if Borrow
  - Memory value is an address
  - Subtract accumulator from value @ address
    - Store result in accumulator & @ address
  - Skip the next location if there’s a borrow
- Location 0 is PC, location 1 is accumulator
  - Can manipulate PC
  - Self-modifying code
- Turing machine – simple FSM, complex tape set-up
- Interesting but not useful(?)

Move y to x:

X
X
X
X
___
___
Y
___
___
X
___

Set x to y-z:

X
X
X
Z
X
___
___
___
___
Y
___
___
X
___
___
___
___

There’s a Whole Flock

- RSSB
- Subtract and branch if negative
  - SUBNEG
  - SUBLEQ
- Subtract
  - And manipulate PC for branches
  - Conditional branch via LUT
- Move
  - 4/8/16 bit data
  - 16/24 bit address
- MaxQ
  - ALU “op codes” as “transfer parameter”
- See ***en.wikipedia.org/siki/OISC***