The programming language of DIY Calculator

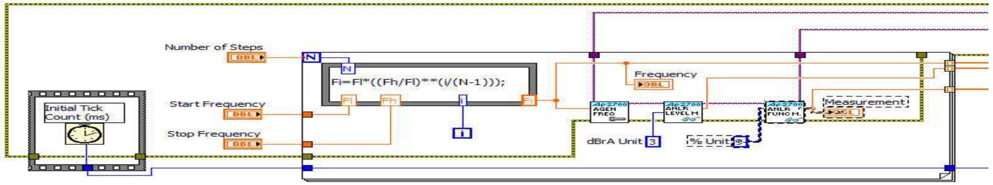
Summarized by Imre Varga

C. Maxfield, A. Brown:
A definitive guide to HOW COMUTERS DO MATH

Programming languages

high level

5GL



```
Fi=Fl*((Fh/Fl)**(i/(N-1)));
```

4GL SELECT name FROM people WHERE age=20

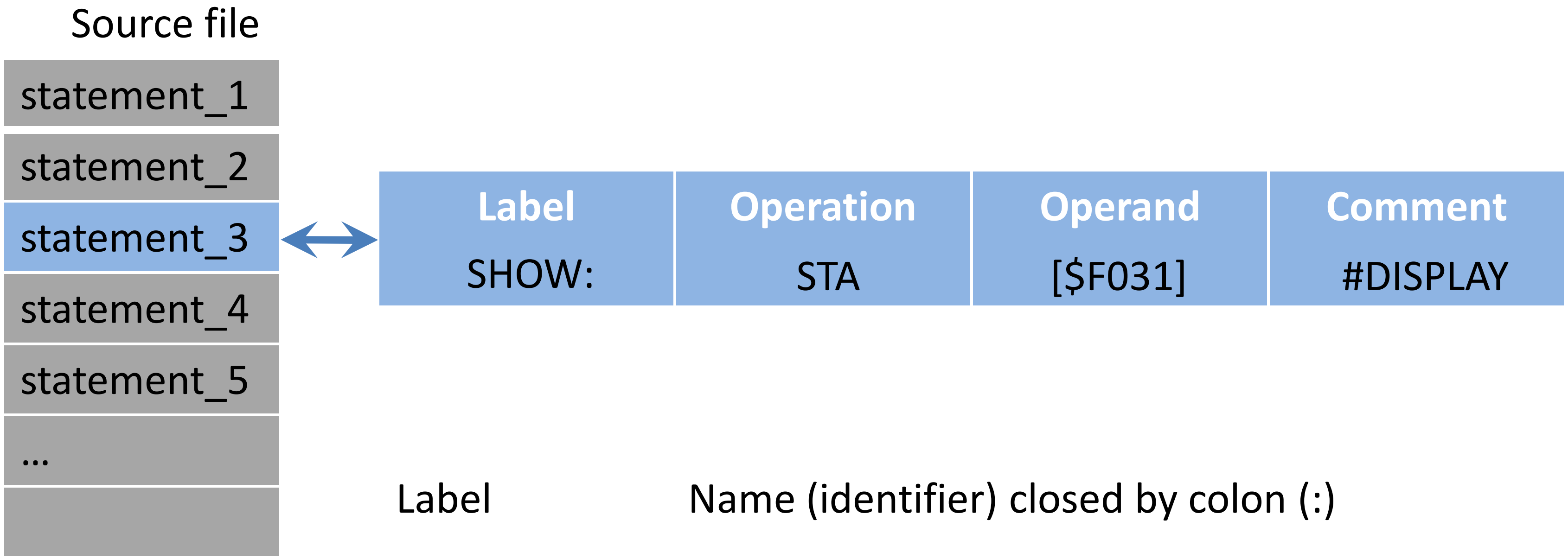
3GL if (x==0) printf(”zero\n”);

2GL SAVE: STA [$410A, X]

1GL 10010110 11101000 10101101 10000111

low level

Source code



Source file

statement_1

statement_2

statement_3

statement_4

statement_5

…

Label	Operation	Operand	Comment
SHOW:	STA	[$F031]	#DISPLAY

Label	Name (identifier) closed by colon (:)
Operation	Instruction mnemonic
Operand	Data (1 byte) or address (2 byte)
Comment	After hash (#) character

Instructions

- Directives
- Load, store
- Bit operation
- ‘Aritmetik-like’
- Control transfer
- Other instructions

Instructions to the assembler

Directives

.EQU	Declare constant value label.
.ORG	Determine the origin of program in the memory.
.BYTE	Reserve 1 byte memory location.
.2BYTE	Reserve 2 byte memory location.
.4BYTE	Reserve 4 byte memory location.
.END	Marks the end of source.

Load & store

Load, store	LDA	Load data in memory into the accumulator.
	STA	Store data in the accumulator into memory.
	BLDX	Load data in memory into the index register.
	BSTX	Store data in the index register into memory.
	BLDSP	Load data in memory into the stack pointer.
	BSTSP	Store data in the stack pointer into memory.
	BLDIV	Load data in memory into the interrupt vector.

Bit operations

Logical	AND	AND data in memory to the accumulator.
	OR	OR data in memory to the accumulator.
	XOR	XOR data in memory to the accumulator.
Shift, rotate	SHL	Shift the accumulator left 1 bit (arithmetic shift).
	SHR	Shift the accumulator right 1 bit (arithmetic shift).
	ROLC	Rotate the accumulator left 1 bit (through carry flag).
	RORC	Rotate the accumulator right 1 bit (through carry flag).

‘Aritmetic-like’

Increment, decrement	INCA	Increment the accumulator.
	DECA	Decrement the accumulator.
	INCX	Increment the index register.
	DECX	Decrement the index register.
Aritmetic	ADD	Add data in memory to the accumulator.
	ADDC	Like an ADD, but include contents of the carry flag.
	SUB	Subtract data in memory from the accumulator.
	SUBC	Like a SUB, but include contents of the carry flag.

Control transfer

Jump	JMP	Jump to a new memory location.
	JSR	Jump to a subroutine.
	JZ	Jump if the result was zero.
	JNZ	Jump if the result wasn’t zero.
	JN	Jump if the result was negative.
	JNN	Jump if the result wasn’t negative.
	JC	Jump if the result generated a carry.
	JNC	Jump if the result didn’t generate a carry.
	JO	Jump if the result generated an overflow.
	JNO	Jump if the result didn’t generate an overflow.
Return	RTS	Return from a subroutine.
	RTI	Return from an interrupt.

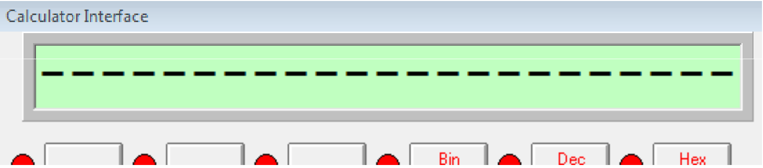
Other instructions

Control	NOP	No-operation, CPU doesn't do anything.
	HALT	Generate internal NOPs until an interrupt occurs.
	SETIM	Set the interrupt mask flag in the status register.
	CLRIM	Clear the interrupt mask flag in the status register.
Comparison	CMPA	Compare data in memory to the accumulator.
Stack	PUSHA	Push the accumulator onto the stack.
	POPA	Pop the accumulator from the stack.
	PUSHSR	Push the status register onto the stack.
	POPSR	Pop the status register from the stack.

Example

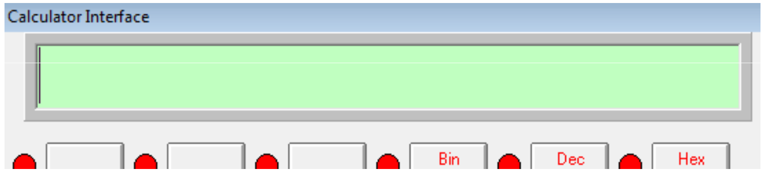
Task:

- *Clear the main display of the front panel!*



```
------------------------
```

LCD display after power on before run



LCD display after the program running

Solution idea:

- *Sending a special value (clearcode) to the LCD display.*

C. Maxfield, A. Brown: A definitive guide to
HOW COMUTERS DO MATH

Questions

- *What is the clearcode?*

  $10
- *What is the address of the LCD display?*

  $F031
- *Where is the clearcode? Where do it have to be?*

  in the accumulator (ACC)
- *How do it get there?*

  with LDA (LoaD Accumulator) instruction
- *How to send value?*

  with STA (Store Accumulator) instruction
- *How does the run finish?*

  with control transfer (JMP instruction) to ROM
- *Do I need other things to determine?*

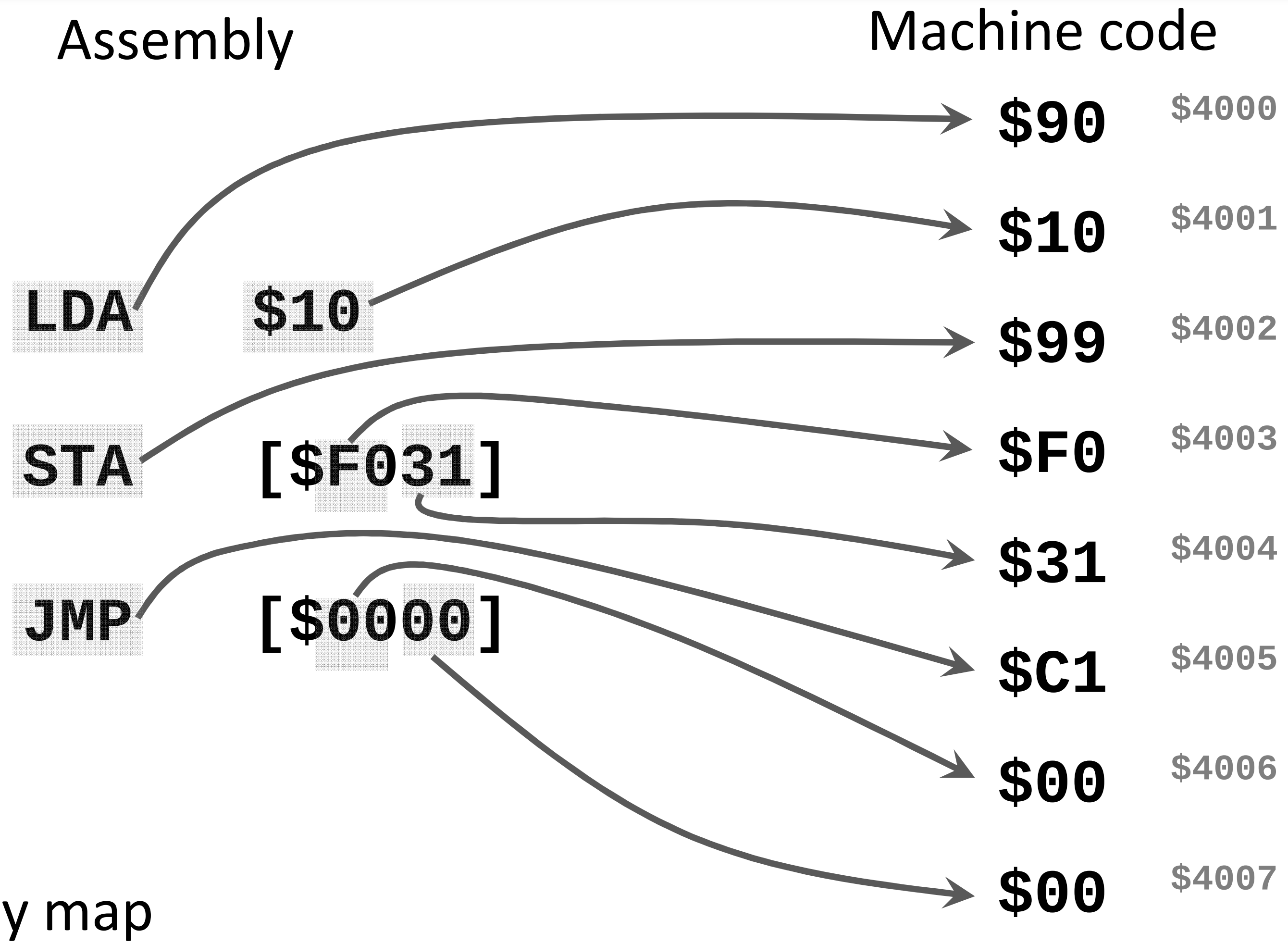
  yes (place of first byre of program in RAM, source end)

Example

Source code:

```
#Clear the main display of the front panel
      .ORG   $4000     #first byte of RAM
      LDA    $10       #load clearcode to ACC
      STA    [$F031]   #store ACC to LCD
      JMP    [$0000]   #control jump to ROM
      .END             #end of source
```

Assembly vs Machine code



Assembly

Machine code

LDA $10

STA [$F031]

JMP [$0000]

$90 $4000

$10 $4001

$99 $4002

$F0 $4003

$31 $4004

$C1 $4005

$00 $4006

$00 $4007

Memory map

1001000000010000100110011111000000110001110000010000000000000000