IxD Theory 2: Telecomunicazioni

IUAV University of Venice

Visual and Multimedia Design graduate programme

Programming the computer 2

What does RobotProg teach us?

The basic elements of his program are:

do something commands

conditional program flow

loops

What does RobotProg teach us?

Programming needs not only knowledge of the language, but also problem-solving strategy

We must make an abstraction of the real-world problem in order to manipulate it with the computer

A programmng language has a limited number of commands that the computer can understand: we have to solve the problem using only these.

Levels of programming languages

Visual programming languages – like Visual Basic or MaxSP

Meta-languages – that write code for you, like Dreamweaver, the web design program

High level languages – like C or Java or Processing

Assembly language – using mnemonics which are then translated into machine language

Machine language (using bit patterns) 0s and 1s

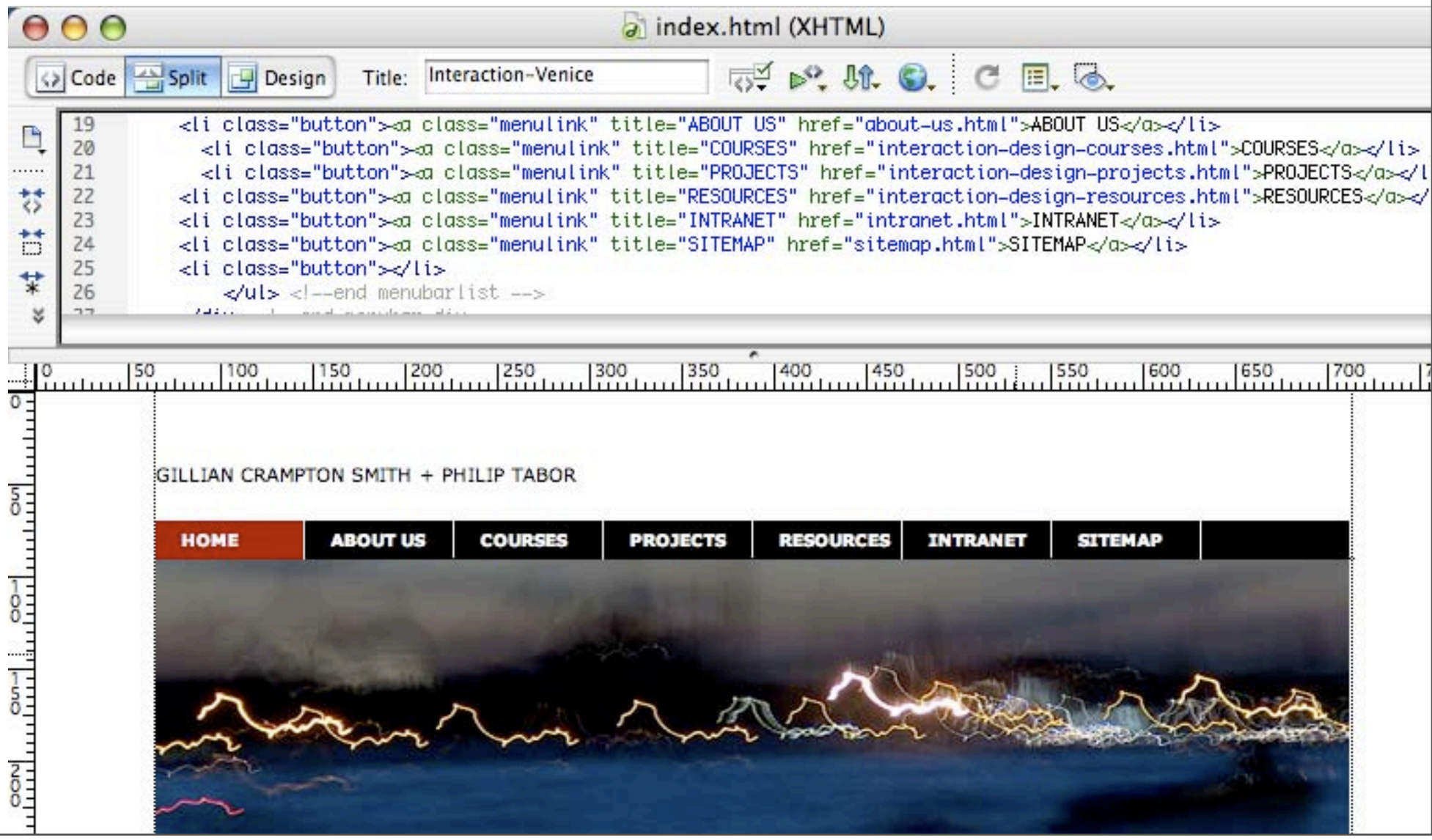
Meta languages

Meta languages

For example: Dreamweaver

- translated first into high-level language (e.g. Html) then into machine code by the computer

- even easier to understand BUT are less flexible.



High-level languages

High-level languages

For example: C++, Java, Processing, HTML and CSS

- are translated into machine code by the computer

- are easier to understand and avoid errors

We still need to know the right commands of our language, but we don't need to worry about which memory locations we are using

```
void setup()
{
size(200, 200);
background(0);
stroke(153);
loop();
}

int x = 0;
int y = 100;
int incx = 1;
int incy = 10;

void draw()
{
line(x, y, x+incx,y+incy);
x = x + incx;
y = y + incy;
if (x + y == 199) noLoop();
}
```

MACHINE LANGUAGE AND ASSEMBLER

How the computer manipulates 0s and 1s

Microprocessor:

- **ALU:** (arithmetic logic unit) which does mathematical operations
- **Registers:** temporary storage locations for operations in progress

Memory (RAM)

- **Longer-term** storage locations
- **Video memory**
- **Screen buffers** special memory mapped to the screen

Machine language

Modern computers use 32-bit numbers to store machine language

000 0001 0010 1011 1000 0000 0010 0000

Means: Add register 1 to register 2, and put the result in register 0.

AAARGH!!!!

Don't panic!

Machine language

At the machine level, the program is written in patterns of 0s and 1s, each of which is a code for things like:

an instruction (e.g. ADD)

a piece of data (e.g. 39)

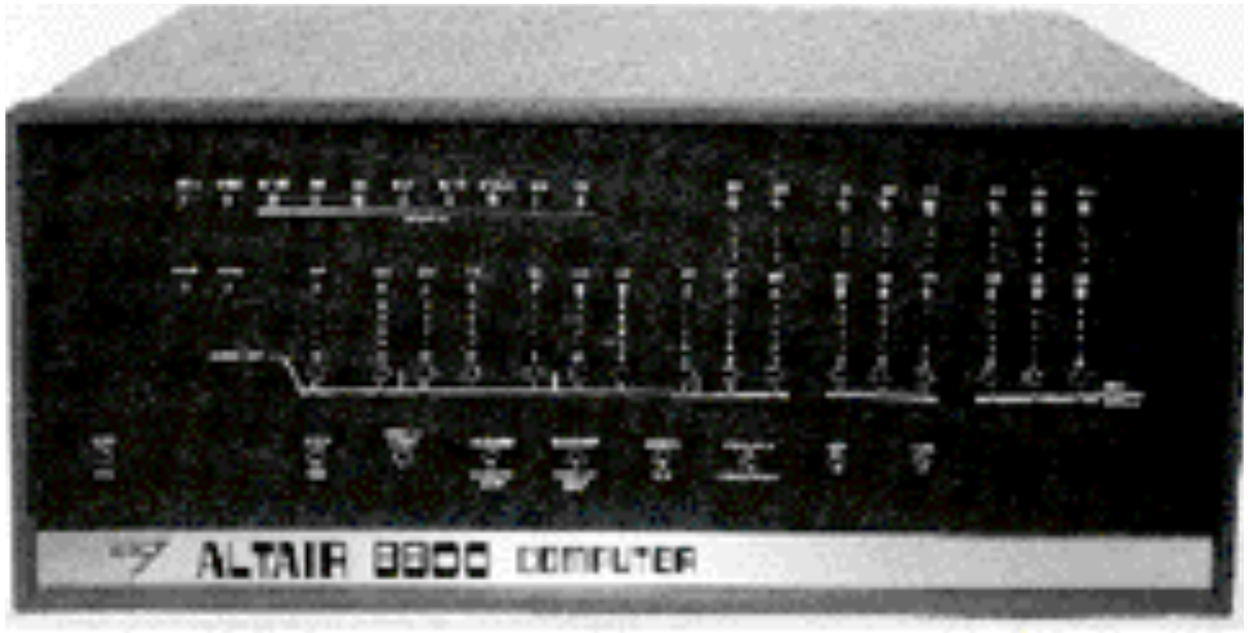
an address in memory to get some data

an address in memory to put some data

Every microprocessor has a different instruction set

Assembly language

The first computers were programmed in machine language.



It was very slow and easy to make errors.

The Altair, an early desktop computer, had two rows of toggle switches: one for the data, and one for the address to put the data.

Assembly language

So assembly languages were invented which used ‘mnemonics’ (easy-to-remember codes). The computer then translated these codes into the 1s and 0s of the machine

A line of assembly language has 4 elements:

Labels	**‘Opcodes’**	**Operand(s)**	**Comments**
SUM:	ADD	$t0, $t1, $t2	;add 2 numbers ;put the result in ;register 2

Assembly language

It’s difficult, but very fast.

Programs take very little memory so it’s good for embedded microprocessors —in white goods, for instance: washing machines, cookers, vacuum cleaners, etc.

http://www.8052.com

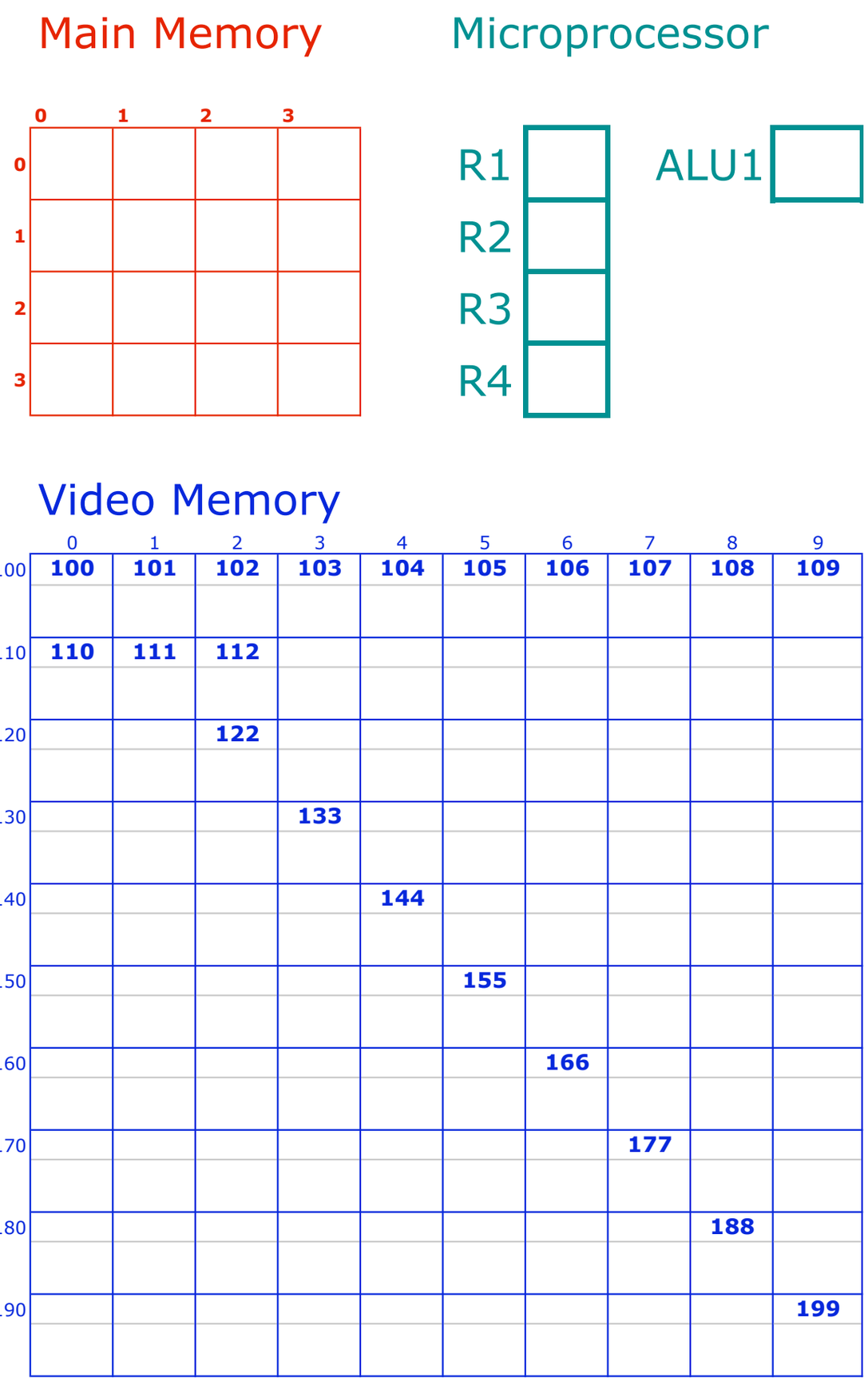
- ACALL: Absolute Call
- ADD, ADDC: Add Accumulator (With Carry)
- AJMP: Absolute Jump
- ANL: Bitwise AND
- CJNE: Compare and Jump if Not Equal
- CLR: Clear Register
- CPL: Complement Register
- DA: Decimal Adjust
- DEC: Decrement Register
- DIV: Divide Accumulator by B
- DJNZ: Decrement Register and Jump if Not Zero
- INC: Increment Register
- JB: Jump if Bit Set
- JBC: Jump if Bit Set and Clear Bit
- JC: Jump if Carry Set
- JMP: Jump to Address
- JNB: Jump if Bit Not Set
- JNC: Jump if Carry Not Set
- JNZ: Jump if Accumulator Not Zero
- JZ: Jump if Accumulator Zero
- LCALL: Long Call
- LJMP: Long Jump
- MOV: Move Memory
- MOVC: Move Code Memory
- MOVX: Move Extended Memory
- MUL: Multiply Accumulator by B
- NOP: No Operation
- ORL: Bitwise OR
- POP: Pop Value From Stack
- PUSH: Push Value Onto Stack
- RET: Return From Subroutine
- RETI: Return From Interrupt
- RL: Rotate Accumulator Left
- RLC: Rotate Accumulator Left Through Carry
- RR: Rotate Accumulator Right
- RRC: Rotate Accumulator Right Through Carry
- SETB: Set Bit
- SJMP: Short Jump
- SUBB: Subtract From Accumulator With Borrow
- SWAP: Swap Accumulator Nibbles
- XCH: Exchange Bytes
- XCHD: Exchange Digits
- XRL: Bitwise Exclusive OR
- Undefined: Undefined Instruction

The Terese micro-computer

Our imaginary low-level computer has only 16 bytes of main memory and 100 bytes of video memory.

Its microprocessor has 4 registers and one arithmetic logic unit.

How would it work?



Assembly language

Remember that Asssembly language commands are like this:

Labels	**'Opcodes'**	**Operand(s)**
SUM:	ADD	$t0, $t1, $t2

So let's look at the command set for our imaginary computer. . . .

Terese micro-computer assember commands

OPCODE	OPERATION		ON operand 1	operand 2	operand 3
LR1	Load Register R1		address or #number	-	-
LR2	Load Register R2		address or #number	-	-
C&M	Copy and move value in 2nd address to 1st address	A	address 1	address 2	
		B	address 1	#number	
ADD	Add 2 values		1st register	2nd register	address to store the result
JMP	jump to a 'label'		label	-	-
JEQ	Checks if two values are equal AND jumps to a label		address 1	address 2	address or 'label' to jump to
INC	Increases a number in address 1 by the number in address 2	A	address 1	address 2	
		B	address 1	#number	
END	Stops the program				

A Terese program

	Label	opcode	operand 1	operand 2	operand 3
1	X:	**C&M**	**0 0**	**#0**	
2	Y:	**C&M**	**0 1**	**#100**	
3	PLUS X:	**C&M**	**0 2**	**#1**	
4	PLUS Y:	**C&M**	**0 3**	**#10**	
5	DRAW:	**LR1**	**0 0**		
6		**LR2**	**0 1**		
7		**ADD**	**R1**	**R2**	**R3**
8		**C&M**	**R3**	**#1**	
9		**INC**	**X**	**0 2**	
10		**INC**	**y**	**0 3**	
11		**JEQ**	**#R3**	**199**	END
12		**JMP**	**DRAW**		
13	END:	**END**			

Assembly language

	Label	opcode	operand 1	operand 2	operand 3	
1	X:	**C&M**	**0 0**	**#0**		;in memory location 0,0 put 0

The Terese micro-computer

Main Memory

	X 0	1	2	3
0	**0**			
1				
2				
3				

Microprocessor

R1 R2 R3 R4 ALU1

	Label	opcode	operand 1	operand 2	operand 3
1	X:	**C&M**	**0 0**	**#0**	

In memory location 0,0 put the number 0

Video Memory

	0	1	2	3	4	5	6	7	8	9
100	**100**	**101**	**102**	**103**	**104**	**105**	**106**	**107**	**108**	**109**
110	**110**	**111**	**112**							
120			**122**							
130				**133**						
140					**144**					
150						**155**				
160							**166**			
170								**177**		
180									**188**	
190										**199**

The Terese micro-computer

;in memory location 0,0 put 0

;in memory location 0,1 put 100

Main Memory

	0	1 (X)	2 (Y)	3
0	0	100		
1				
2				
3				

Microprocessor

R1 R2 R3 R4

ALU1

Video Memory

	0	1	2	3	4	5	6	7	8	9
100	100	101	102	103	104	105	106	107	108	109
110	110	111	112							
120			122							
130				133						
140					144					
150						155				
160							166			
170								177		
180									188	
190										199

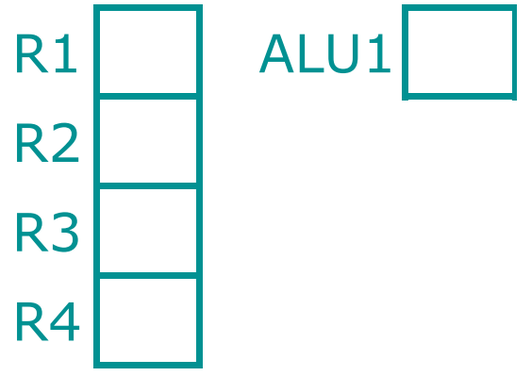
The Terese micro-computer

```
;in memory location 0,0 put 0
;in memory location 0,1 put 100
;in memory location 0,2 put 1
;in memory location 0,3 put 10
```

Main Memory

	0 X	1 Y	2 X+	3 Y+
0	0	100	1	10
1				
2				
3				

Microprocessor



Video Memory

	0	1	2	3	4	5	6	7	8	9
100	100	101	102	103	104	105	106	107	108	109
110	110	111	112							
120			122							
130				133						
140					144					
150						155				
160							166			
170								177		
180									188	
190										199

The Terese micro-computer

;in memory location 0,0 put 0
;in memory location 0,1 put 100
;in memory location 0,2 put 1
;in memory location 0,3 put 10

; load R1 with contents of memory 00

Main Memory

	0 X	1 Y	2 X+	3 Y+
0	**0**	100	1	10
1				
2				
3				

Microprocessor

R1	**0**
R2	
R3	
R4	

ALU1

Video Memory

	0	1	2	3	4	5	6	7	8	9
100	**100**	**101**	**102**	**103**	**104**	**105**	**106**	**107**	**108**	**109**
110	**110**	**111**	**112**							
120			**122**							
130				**133**						
140					**144**					
150						**155**				
160							**166**			
170								**177**		
180									**188**	
190										**199**

The Terese micro-computer

;in memory location 0,0 put 0
;in memory location 0,1 put 100
;in memory location 0,2 put 1
;in memory location 0,3 put 10

; load R1 with contents of memory 00

; load R2 with contents of memory 01

Main Memory

	0 X	1 Y	2 X+	3 Y+
0	**0**	**100**	**1**	**10**
1				
2				
3				

Microprocessor

R1	**0**
R2	**100**
R3	
R4	

ALU1

Video Memory

	0	1	2	3	4	5	6	7	8	9
100	**100**	**101**	**102**	**103**	**104**	**105**	**106**	**107**	**108**	**109**
110	**110**	**111**	**112**							
120			**122**							
130				**133**						
140					**144**					
150						**155**				
160							**166**			
170								**177**		
180									**188**	
190										**199**

The Terese micro-computer

;in memory location 0,0 put 0
;in memory location 0,1 put 100
;in memory location 0,2 put 1
;in memory location 0,3 put 10

; load R1 with contents of memory 00
; load R2 with contents of memory 01
;add

Main Memory

	0 X	1 Y	2 X+	3 Y+
0	0	100	1	10
1				
2				
3				

Microprocessor

R1	0
R2	100
R3	
R4	

ALU1 0+ 100

Video Memory

	0	1	2	3	4	5	6	7	8	9
100	100	101	102	103	104	105	106	107	108	109
110	110	111	112							
120			122							
130				133						
140					144					
150						155				
160							166			
170								177		
180									188	
190										199

The Terese micro-computer

;in memory location 0,0 put 0
;in memory location 0,1 put 100
;in memory location 0,2 put 1
;in memory location 0,3 put 10

; load R1 with contents of memory 00

; load R2 with contents of memory 01

;add and put result in R3

Main Memory

	0 X	1 Y	2 X+	3 Y+
0	0	100	1	10
1				
2				
3				

Microprocessor

R1	0
R2	100
R3	100
R4	

ALU1: 0+ 100

Video Memory

	0	1	2	3	4	5	6	7	8	9
100	100	101	102	103	104	105	106	107	108	109
110	110	111	112							
120			122							
130				133						
140					144					
150						155				
160							166			
170								177		
180									188	
190										199

The Terese micro-computer

;in memory location 0,0 put 0
;in memory location 0,1 put 100
;in memory location 0,2 put 1
;in memory location 0,3 put 10

; load R1 with contents of memory 00
; load R2 with contents of memory 01
;add and put result in R3
; put 1 (to light a pixel) in the address in R3

Main Memory

	0 X	1 Y	2 X+	3 Y+
0	**0**	**100**	**1**	**10**
1				
2				
3				

Microprocessor

R1	**0**
R2	**100**
R3	**100**
R4	

ALU1

Video Memory

	0	1	2	3	4	5	6	7	8	9
100		**101**	**102**	**103**	**104**	**105**	**106**	**107**	**108**	**109**
110	**110**	**111**	**112**							
120			**122**							
130				**133**						
140					**144**					
150						**155**				
160							**166**			
170								**177**		
180									**188**	
190										**199**

The Terese micro-computer

;in memory location 0,0 put 0
;in memory location 0,1 put 100
;in memory location 0,2 put 1
;in memory location 0,3 put 10

; load R1 with contents of memory 00
; load R2 with contents of memory 01
;add and put result in R3
; put 1 (to light a pixel) in the address in R3

; increment x with contents of memory 02

Main Memory

	0 X	1 Y	2 X+	3 Y+
0	**0**	**100**	**1**	**10**
1				
2				
3				

Microprocessor

R1	**0**
R2	**100**
R3	**100**
R4	

ALU1

Video Memory

	0	1	2	3	4	5	6	7	8	9
100		**101**	**102**	**103**	**104**	**105**	**106**	**107**	**108**	**109**
110	**110**	**111**	**112**							
120			**122**							
130				**133**						
140					**144**					
150						**155**				
160							**166**			
170								**177**		
180									**188**	
190										**199**

The Terese micro-computer

;in memory location 0,0 put 0
;in memory location 0,1 put 100
;in memory location 0,2 put 1
;in memory location 0,3 put 10

; load R1 with contents of memory 00
; load R2 with contents of memory 01
;add and put result in R3
; put 1 (to light a pixel) in the address in R3

; increment x with contents of memory 02

Main Memory

	0 X	1 Y	2 X+	3 Y+
0	**1**	**100**	**1**	**10**
1				
2				
3				

Microprocessor

R1	**0**
R2	**100**
R3	**100**
R4	

ALU1

Video Memory

	0	1	2	3	4	5	6	7	8	9
100		**101**	**102**	**103**	**104**	**105**	**106**	**107**	**108**	**109**
110	**110**	**111**	**112**							
120			**122**							
130				**133**						
140					**144**					
150						**155**				
160							**166**			
170								**177**		
180									**188**	
190										**199**

The Terese micro-computer

;in memory location 0,0 put 0
;in memory location 0,1 put 100
;in memory location 0,2 put 1
;in memory location 0,3 put 10

; load R1 with contents of memory 00
; load R2 with contents of memory 01
;add and put result in R3
; put 1 (to light a pixel) in the address in R3

; increment x with contents of memory 02
; increment y with contents of memory 03

Main Memory

	0 X	1 Y	2 X+	3 Y+
0	0	**100**	1	**10**
1				
2				
3				

Microprocessor

R1	0
R2	100
R3	100
R4	

ALU1

Video Memory

	0	1	2	3	4	5	6	7	8	9
100		**101**	**102**	**103**	**104**	**105**	**106**	**107**	**108**	**109**
110	**110**	**111**	**112**							
120			**122**							
130				**133**						
140					**144**					
150						**155**				
160							**166**			
170								**177**		
180									**188**	
190										**199**

The Terese micro-computer

;in memory location 0,0 put 0
;in memory location 0,1 put 100
;in memory location 0,2 put 1
;in memory location 0,3 put 10

; load R1 with contents of memory 00
; load R2 with contents of memory 01
;add and put result in R3
; put 1 (to light a pixel) in the address in R3

; increment x with contents of memory 02
; increment y with contents of memory 03

Main Memory

	0 X	1 Y	2 X+	3 Y+
0	0	**110**	1	**10**
1				
2				
3				

Microprocessor

R1	0
R2	100
R3	100
R4	

ALU1

Video Memory

	0	1	2	3	4	5	6	7	8	9
100		**101**	**102**	**103**	**104**	**105**	**106**	**107**	**108**	**109**
110	**110**	**111**	**112**							
120			**122**							
130				**133**						
140					**144**					
150						**155**				
160							**166**			
170								**177**		
180									**188**	
190										**199**

The Terese micro-computer

Main Memory

	0 X	1 Y	2 X+	3 Y+
0	**1**	**110**	**1**	**10**
1				
2				
3				

Microprocessor

R1	**0**
R2	**100**
R3	**100**
R4	

ALU1

Video Memory

	0	1	2	3	4	5	6	7	8	9
100		**101**	**102**	**103**	**104**	**105**	**106**	**107**	**108**	**109**
110	**110**	**111**	**112**							
120			**122**							
130				**133**						
140					**144**					
150						**155**				
160							**166**			
170								**177**		
180									**188**	
190										**199**

;in memory location 0,0 put 0

;in memory location 0,1 put 100

;in memory location 0,2 put 1

;in memory location 0,3 put 10

; load R1 with contents of memory 00

; load R2 with contents of memory 01

;add and put result in R3

; put 1 (to light a pixel) in the address in R3

; increment x with contents of memory 02

; increment y with contents of memory 03

; check if R3=199 (R3 is where the video memory location was stored);

; jump to label 'DRAW'

The Terese micro-computer

```
        ;in memory location 0,0 put 0
        ;in memory location 0,1 put 100
        ;in memory location 0,2 put 1
        ;in memory location 0,3 put 10
DRAW    ; load R1 with contents of
        memory 00
        ; load R2 with contents of
        memory 01
        ;add and put result in R3
        ; put 1 (to light a pixel) in the
        address in R3
```

Main Memory

	0 X	1 Y	2 X+	3 Y+
0	**1**	**110**	**1**	**10**
1				
2				
3				

Microprocessor

R1	**1**
R2	**110**
R3	**111**
R4	

ALU1

Video Memory

	0	1	2	3	4	5	6	7	8	9
100		**101**	**102**	**103**	**104**	**105**	**106**	**107**	**108**	**109**
110	**110**		**112**							
120			**122**							
130				**133**						
140					**144**					
150						**155**				
160							**166**			
170								**177**		
180									**188**	
190										**199**

The Terese micro-computer

```
          ;in memory location 0,0 put 0
          ;in memory location 0,1 put 100
          ;in memory location 0,2 put 1
          ;in memory location 0,3 put 10
DRAW      ; load R1 with contents of
          memory 00
          ; load R2 with contents of
          memory 01
          ;add and put result in R3
          ; put 1 (to light a pixel) in the
          address in R3
          ; increment x with contents of
          memory 02
          ; increment y with contents of
          memory 03
          ; check if R3=199 (R3 is where
          the video memory location was
          stored);
          ; jump to label 'DRAW'
```

Main Memory

	0 X	1 Y	2 X+	3 Y+
0	**2**	**120**	**1**	**10**
1				
2				
3				

Microprocessor

R1	**1**
R2	**110**
R3	**111**
R4	

ALU1

Video Memory

	0	1	2	3	4	5	6	7	8	9
100		**101**	**102**	**103**	**104**	**105**	**106**	**107**	**108**	**109**
110	**110**		**112**							
120			**122**							
130				**133**						
140					**144**					
150						**155**				
160							**166**			
170								**177**		
180									**188**	
190										**199**

The Terese micro-computer

;in memory location 0,0 put 0
;in memory location 0,1 put 100
;in memory location 0,2 put 1
;in memory location 0,3 put 10

; load R1 with contents of memory 00
; load R2 with contents of memory 01
;add and put result in R3
; put 1 (to light a pixel) in the address in R3

Main Memory

	0 X	1 Y	2 X+	3 Y+
0	**2**	**120**	**1**	**10**
1				
2				
3				

Microprocessor

R1	**2**
R2	**120**
R3	**122**
R4	

ALU1

Video Memory

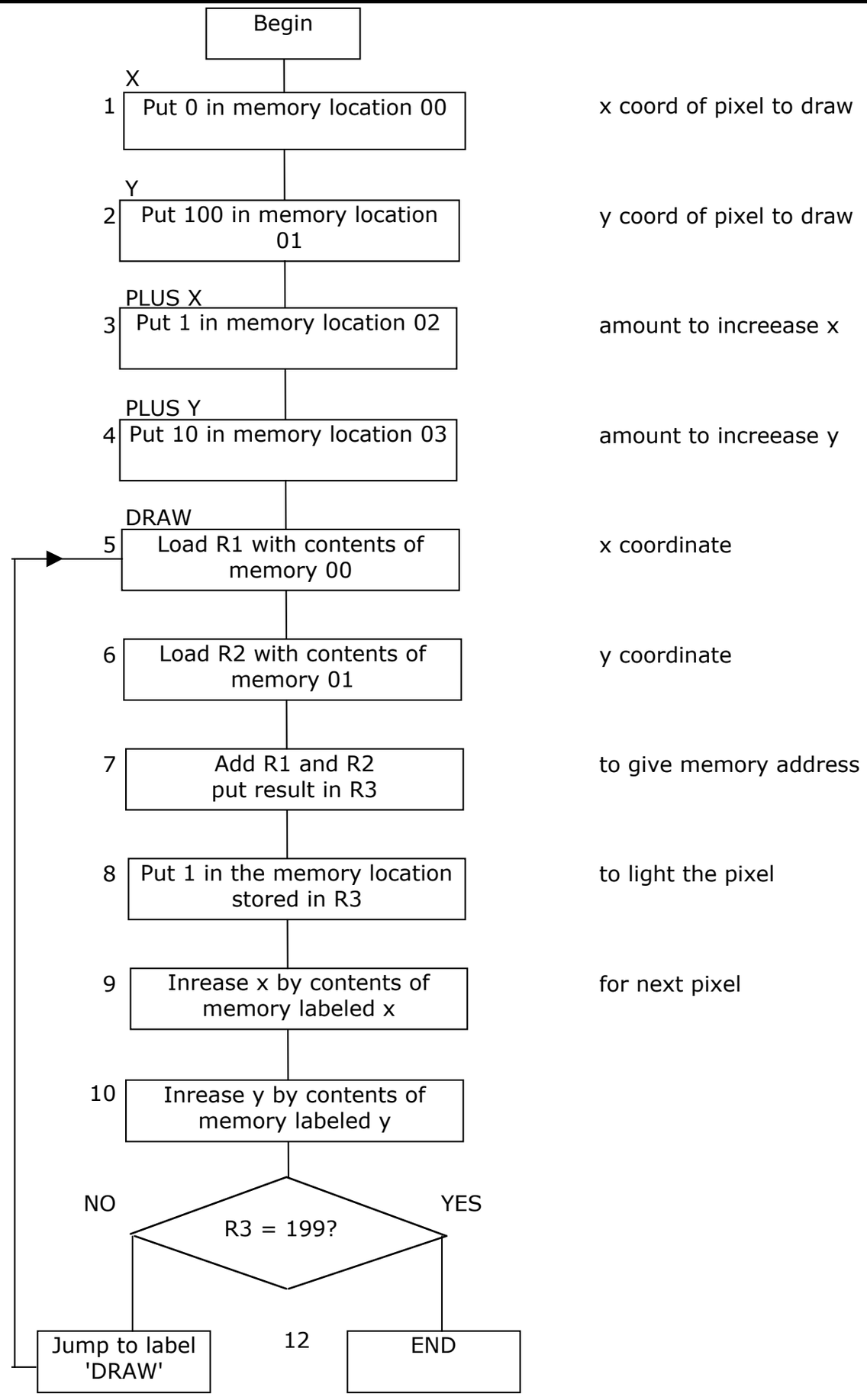
	0	1	2	3	4	5	6	7	8	9
100		**101**	**102**	**103**	**104**	**105**	**106**	**107**	**108**	**109**
110	**110**		**112**							
120										
130				**133**						
140					**144**					
150						**155**				
160							**166**			
170								**177**		
180									**188**	
190										**199**

Assembly language

Terese mini-computer: example of assembler program

	Label	opcode	operand 1	operand 2	operand 3	
1	X:	**C&M**	**0 0**	**#0**		;in memory location 0,0 put 0
2	Y:	**C&M**	**0 1**	**#100**		;in memory location 0,1 put 100
3	PLUS X:	**C&M**	**0 2**	**#1**		;in memory location 0,2 put 1
4	PLUS Y:	**C&M**	**0 3**	**#10**		;in memory location 0,3 put 10
5	DRAW:	**LR1**	**0 0**			; load R1 with contents of memory 00
6		**LR2**	**0 1**			; load R2 with contents of memory 01
7		**ADD**	**R1**	**R2**	**R3**	;add and put result in R3
8		**C&M**	**R3**	**#1**		; put 1 (to light a pixel) in the address in R3
9		**INC**	**X**	**0 2**		; increment x with contents of memory 02
10		**INC**	**y**	**0 3**		; increment y with contents of memory 03
11		**JEQ**	**#R3**	**199**	END	; check if R3=199 (R3 is where the video memory location was stored);
12		**JMP**	**DRAW**			; jump to label 'DRAW'
13	END:	**END**				

Flowchart


Begin
X
1
Put 0 in memory location 00
x coord of pixel to draw
Y
2
Put 100 in memory location 01
y coord of pixel to draw
PLUS X
3
Put 1 in memory location 02
amount to increease x
PLUS Y
4
Put 10 in memory location 03
amount to increease y
DRAW
5
Load R1 with contents of memory 00
x coordinate
6
Load R2 with contents of memory 01
y coordinate
7
Add R1 and R2 put result in R3
to give memory address
8
Put 1 in the memory location stored in R3
to light the pixel
9
Inrease x by contents of memory labeled x
for next pixel
10
Inrease y by contents of memory labeled y
NO
R3 = 199?
YES
Jump to label 'DRAW'
12
END