

FPGA IMPLEMENTATION OF A RISC-V PROCESSOR AND CUSTOM OS DEVELOPMENT

Intel MAX 10 FPGA / Terasic DE10-Lite board

Saba Prangulashvili

GOAL

====

Build a complete small computer on an FPGA, layer by layer:

+	-----	+
	SOFTWARE kernel + command shell in RISC-V assembly	
+	-----	+
	I/O VGA text display + PS/2 keyboard	
+	-----	+
	CPU single-cycle RISC-V processor	
+	-----	+

Every layer written from scratch, no ready-made CPU core

IMPLEMENTATION

=====

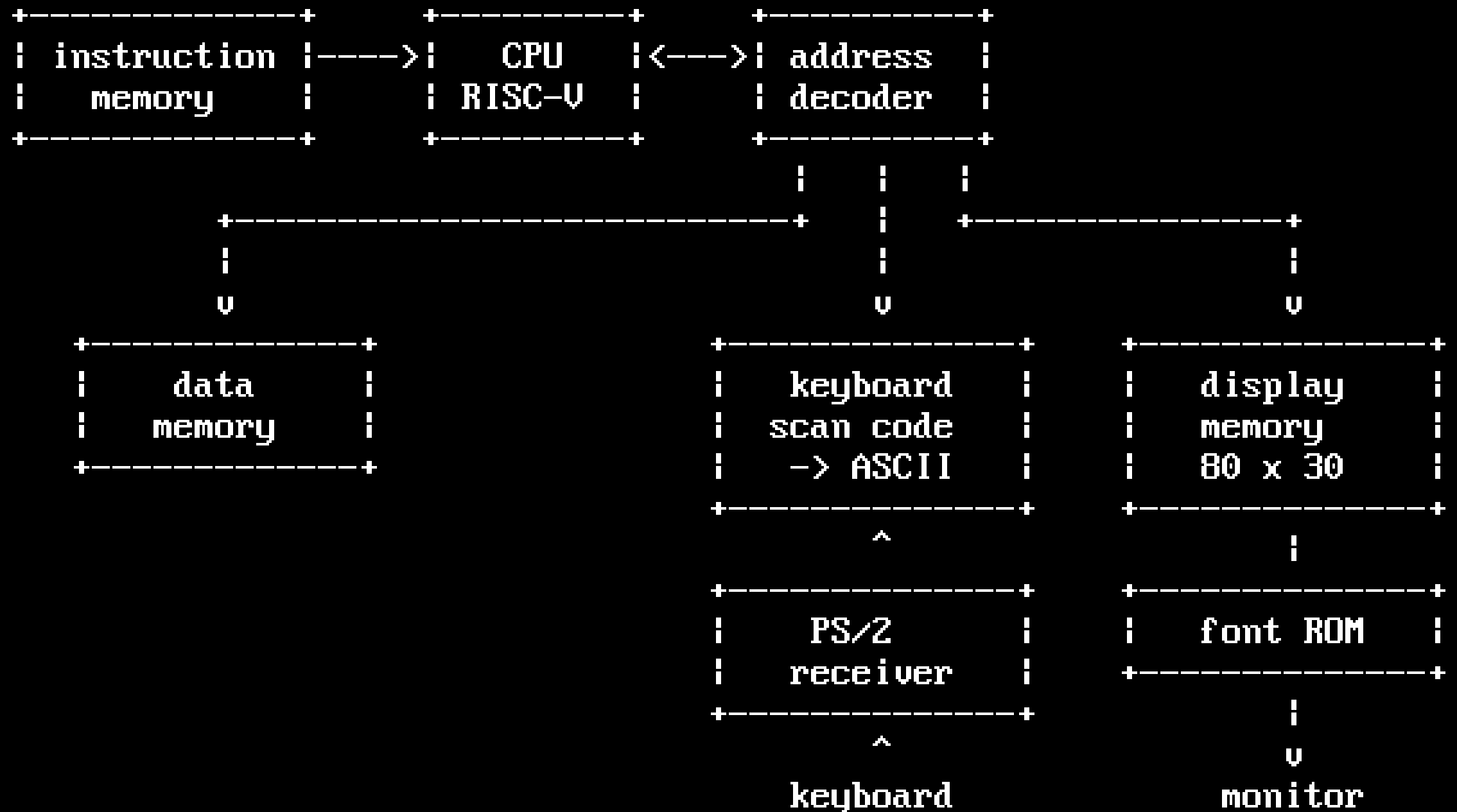
- 1 CPU ALU, register file, decoders, datapath
 -> each module tested in simulation
- 2 Display VGA 640 x 480 timing, 80 x 30 text mode, font ROM
- 3 Keyboard PS/2 receiver, scan code -> ASCII
- 4 System memory-mapped I/O: RAM / keyboard / display
- 5 Software own assembler (Python), kernel, drivers, shell

Tools SystemVerilog, Quartus, Icarus Verilog, Python

Reference Harris & Harris, "Digital Design and Computer
 Architecture: RISC-V Edition"

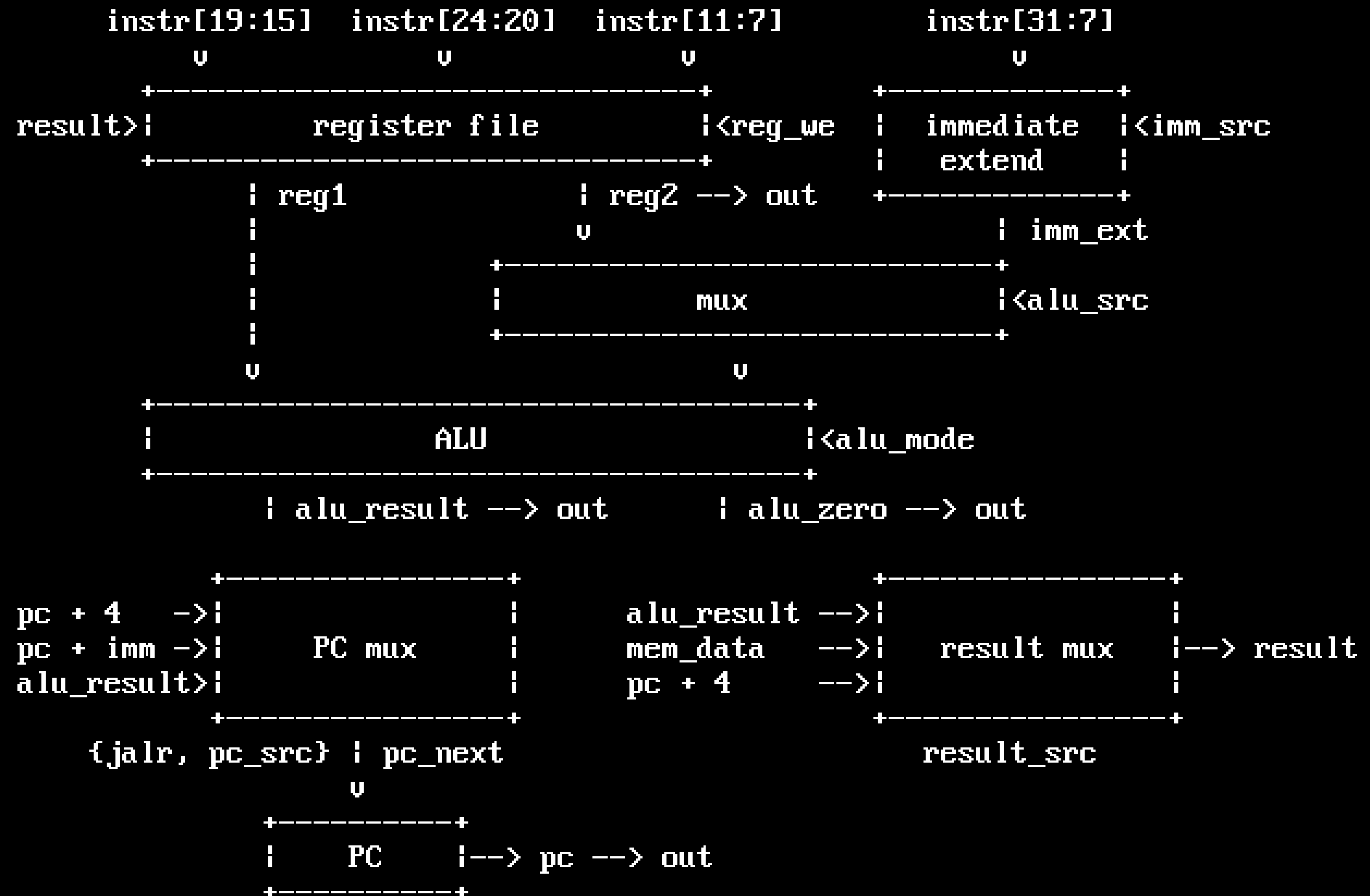
SYSTEM

=====



DATA PATH

=====



INSTRUCTION SET

=====

type	instructions	example
R	add sub and or slt	add x3, x1, x2
I	addi andi ori slti	addi x3, x1, 5
load	lw	lw x6, 8(x2)
store	sw	sw x6, 8(x2)
branch	beq	beq x1, x2, loop
jump	jal jalr	jal x1, func

	31	25 24	20 19	15 14	12 11	7 6	0
R	funct7	rs2	rs1	f3	rd	opcode	
I	imm[11:0]	rs1	f3	rd	opcode		
S	imm	rs2	rs1	f3	imm	opcode	
B	imm	rs2	rs1	f3	imm	opcode	
J		imm			rd	opcode	

CONTROL UNIT

=====

```

opcode -----> [ main decoder ] --> control signals
                |
                +-- alu_op
                |
                v
funct3, funct7 -----> [ ALU decoder ] --> ALU operation
  
```

instr	reg write	mem write	ALU input B	result	next PC
-----	-----	-----	-----	-----	-----
R-type	yes	-	register	ALU	pc + 4
I-type	yes	-	immediate	ALU	pc + 4
lw	yes	-	immediate	memory	pc + 4
sw	-	yes	immediate	-	pc + 4
beq	-	-	register	-	pc + imm if equal
jal	yes	-	-	pc + 4	pc + imm
jalr	yes	-	immediate	pc + 4	rs1 + imm

ALU operations: add sub and or slt

MEMORY MAP

=====

CPU address space		
+-----+	0x0000	
data memory		
+-----+		
keyboard	0x2000	lw -> ASCII of last key
+-----+		
(unused)		
+-----+	0x3000	
display memory		sw -> character
80 x 30 = 2400		on screen
+-----+	0x395F	

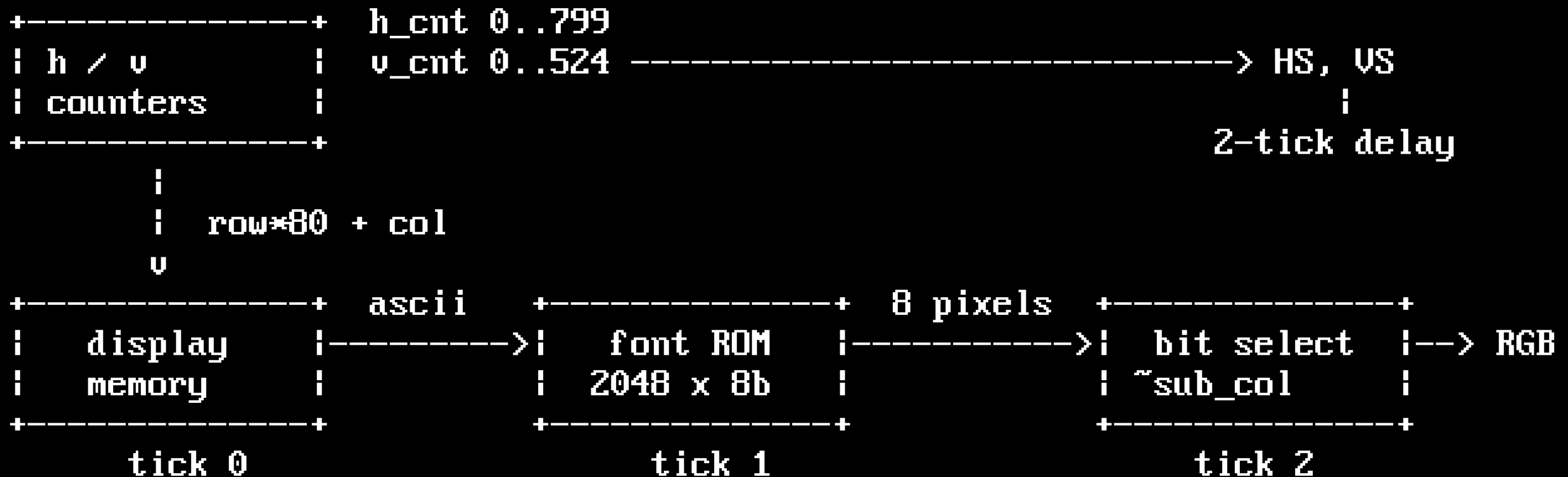
screen cell = row * 80 + column

sw x10, 81(x3) x3 = 0x3000 -> row 1, column 1

DISPLAY: 640 x 480 @ 60 Hz TEXT MODE

=====

50 MHz --> /2 --> pixel_en 25 MHz



|<----- 640 visible ----->|<16>|<-- 96 -->|<48>|
front HSYNC back

cell = 8 x 16 px

grid = 80 x 30

font = IBM VGA 8x16

KEYBOARD: PS/2

=====

clock	level	PS/2	scan code	CPU
PS/2 keyboard	shifter	receiver	-> ASCII	
data	5V->3.3V			
				lw 0x2000

press A --> 1C --> 'A'

release A --> F0 1C --> ignored

SOFTWARE

=====

shell.s ---> assembler.py ---> main.txt ---> instruction memory

```
+-----+
|  shell      prompt, line buffer, clr / help / echo      |
+-----+
|  library    strcmp  div  print_str  disp_scroll          |
+-----+
|  drivers    print_char  disp_clear  getc                  |
+-----+
|  kernel     set base registers, build strings, run shell |
+-----+
|  hardware   CPU  +  memory  +  keyboard  +  display      |
+-----+
```

fixed registers	x2	stack pointer	
	x3	display memory base	0x3000
	x4	keyboard address	0x2000
calls	jal x1, func	...	jalr x0, 0(x1)
arguments / return	x10, x11		

```
> help
clr, echo, help
> echo HELLO WORLD
HELLO WORLD
> ls
Wrong command
> _
```

RESULTS

=====

- single-cycle RISC-V CPU, running 13 RV32I instructions including function calls and returns
- VGA text display: 80 x 30 characters on a 640 x 480 monitor
- PS/2 keyboard input: keys typed on a real keyboard appear on screen
- memory-mapped I/O connecting CPU, memory, keyboard and display
- own assembler, kernel and command shell (help, echo, clr)
- slide presentation running on the computer

NEXT

====

[1	full RV32I	bne blt bge lui auipc shifts lb sb ...
[1	C programs	riscv32-unknown-elf-gcc
[1	M extension	mul div
[1	pipelined CPU	IF ID EX MEM WB
[1	interrupts	timer, keyboard IRQ instead of polling