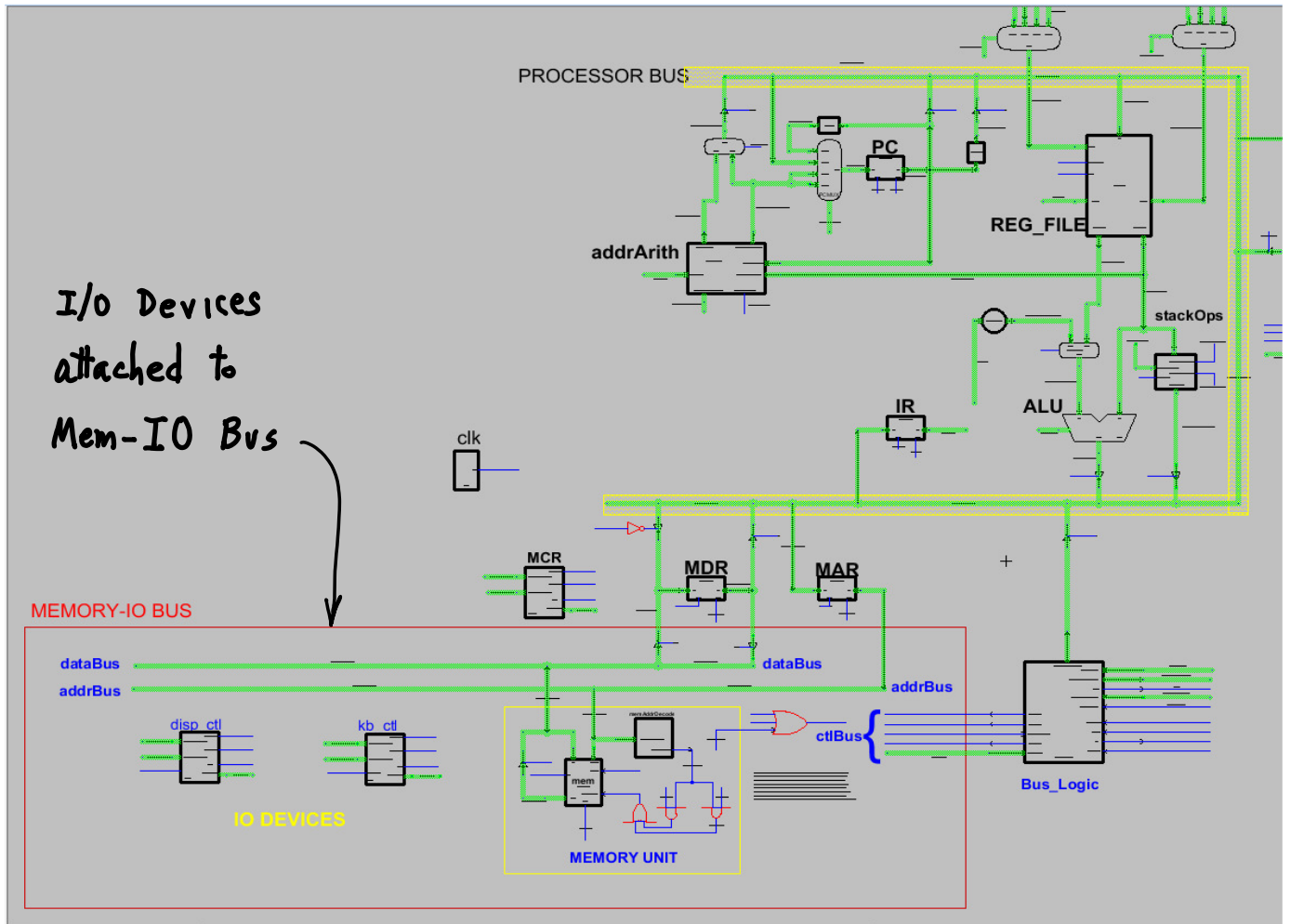
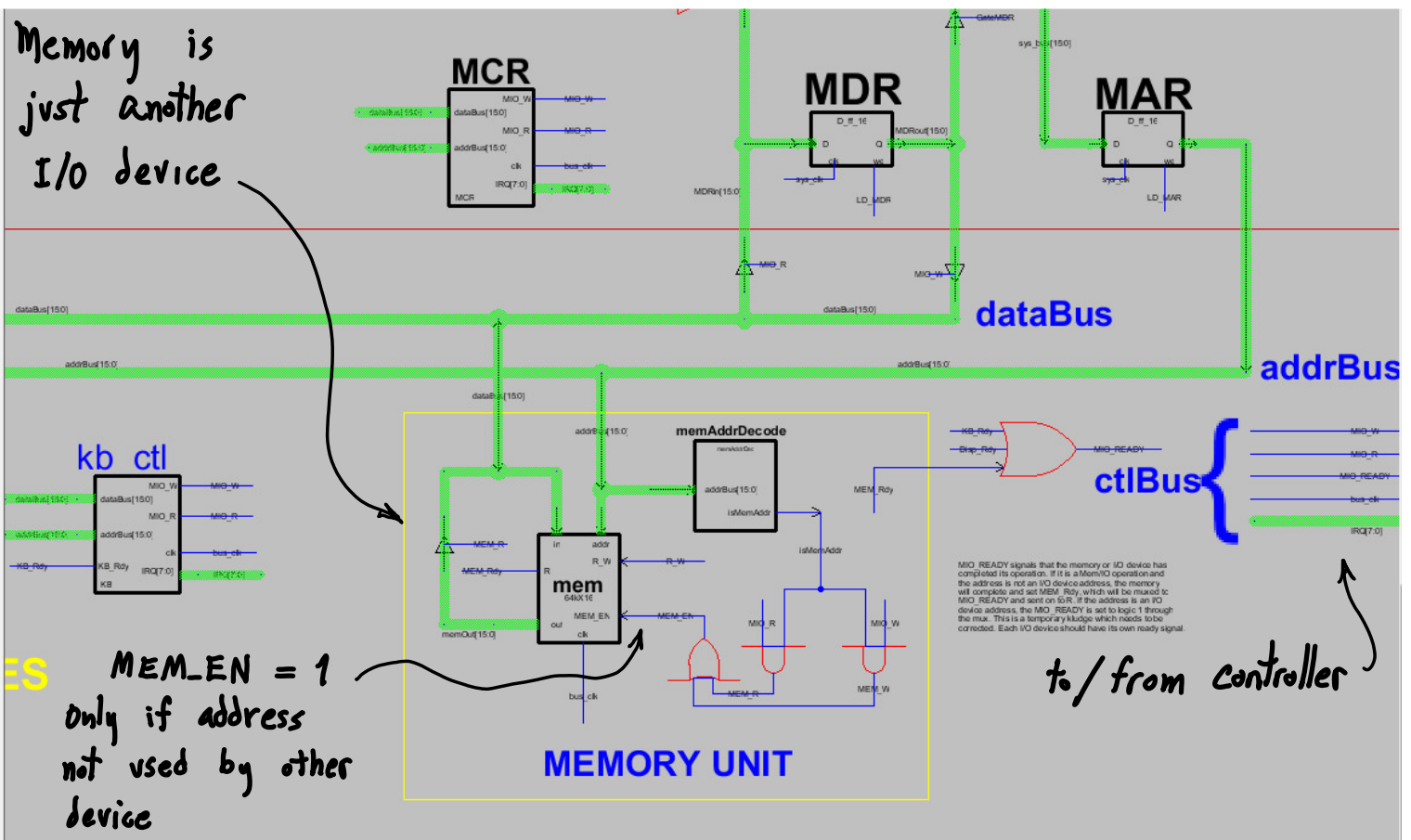


# Programming Input/Output



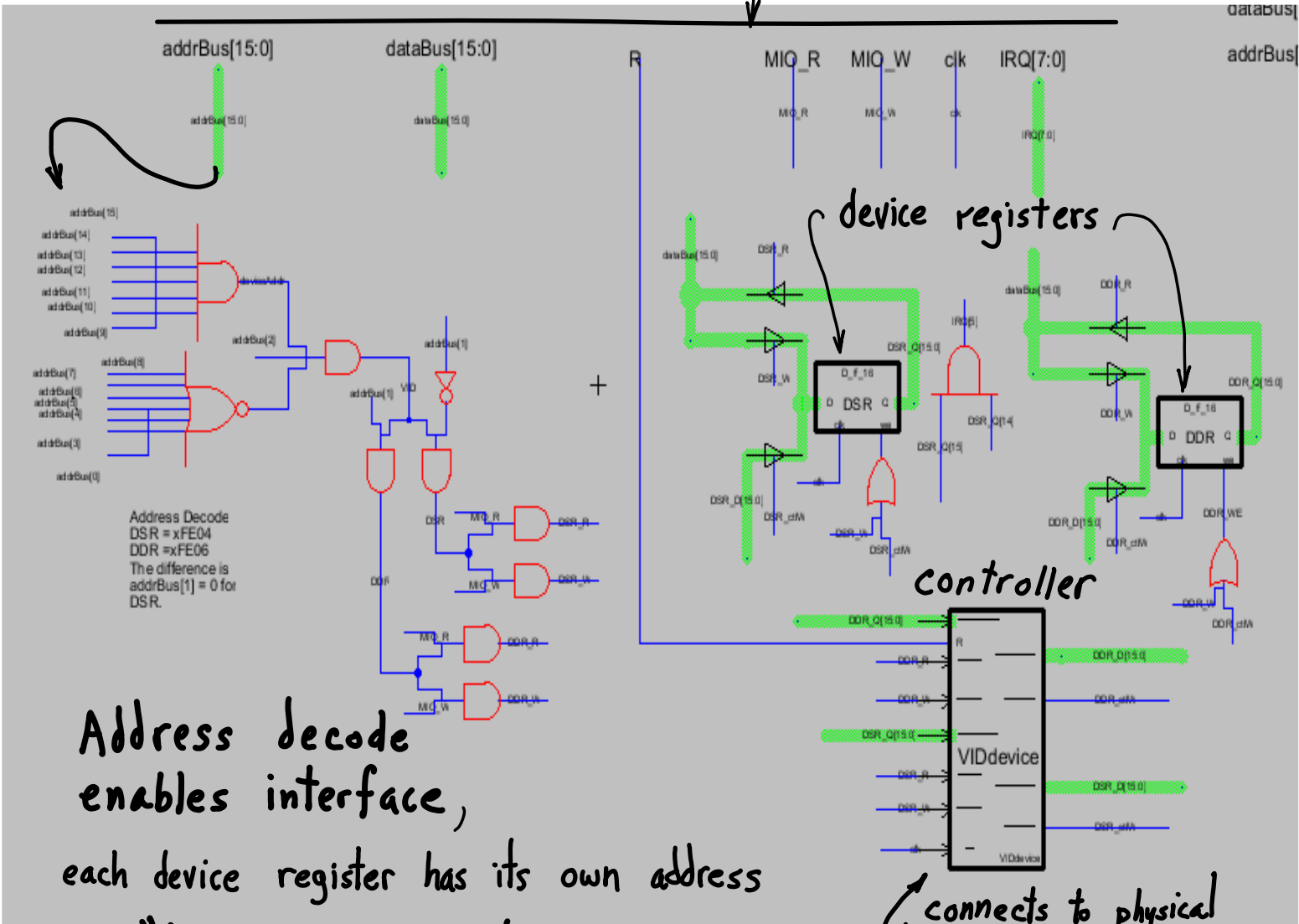
Memory is just another I/O device



MIO\_READY signals that the memory or IO device has completed its operation. If it is a Mem/IO operation and the address is not an IO device address, the memory will complete and set MEM\_Rdy which will be muxed to MIO\_READY and sent on IO\_R. If the address is an IO device address, the MIO\_READY is set to logic 1 through the mux. This is a temporary fudge which needs to be corrected. Each I/O device should have its own ready signal.

# Display ctl, device controller

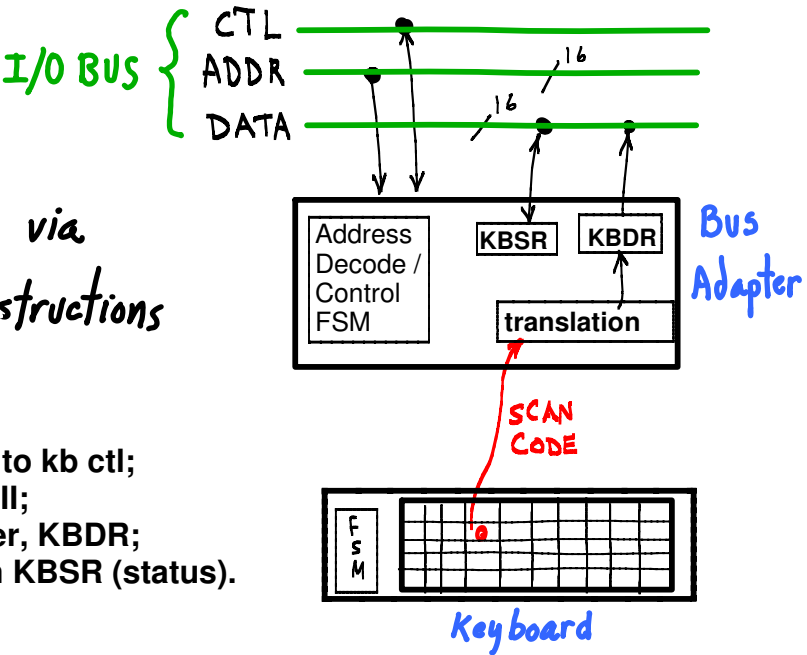
I/O Bus Connections



Address decode enables interface,  
 each device register has its own address  
 → "memory-mapped I/O"

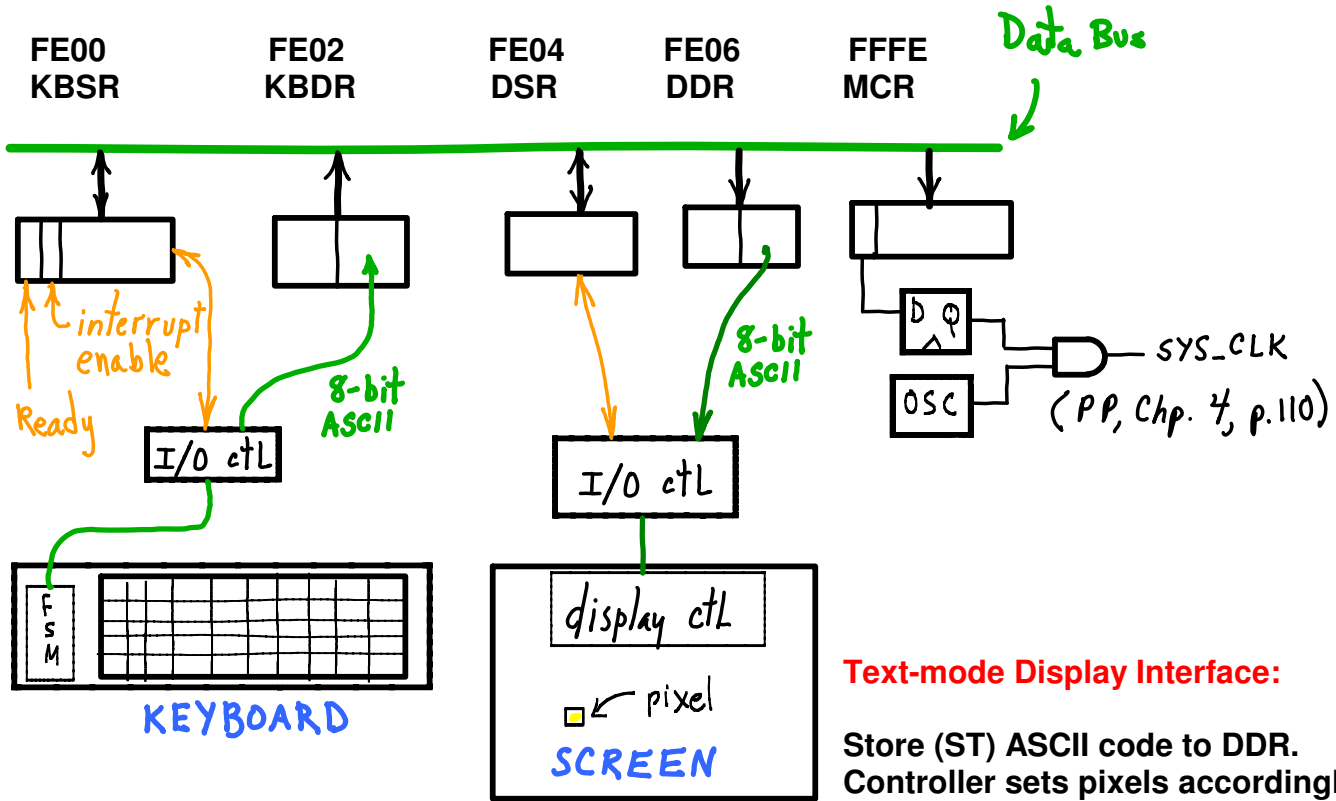
connects to physical display hardware

addr	mem
0000:	Trap vector
...	...
01FF:	Interrupt vector
0200:	OS code/text
...	...
...	OS data
...	OS heap
...	OS stack
2FFF:	USER code/text
3000:	USER data
...	...
...	USER heap
...	USER stack
FDFF:	USER stack
FE00:	Device register
...	...
FFFF:	Device register



Memory address decoder does not recognize these memory addresses.

Physical memory locations for addresses FE00 - FFFF are not accessible.



```

;-----
;--- keyboard access
;-----
.ORIG x3000

```

```

LDI R0, KBDRptr
BRnzp DONE

```

```

KBDRptr:
.FILL xFE02

```

```

DONE:
;-----
;--- Execution:
;--- R0 <== MEM[ MEM[ 3002 ] ]
;---      == MEM[ xFE02 ]
;---      == KBDR

```

### Graphics-mode Display Interface (PennSim):

Each pixel controlled by individual memory address:

range == C000 - FFDF

Pixel at position	is controlled by location
(0, 0)	C000
(0, 1)	C001
...	...
(0, 127)	C07F
(1, 0)	C080
(1, 1)	C081
...	...
(123, 127)	FDFF

Pixel code == [ 0 RRRRR BBBB GGGG ]

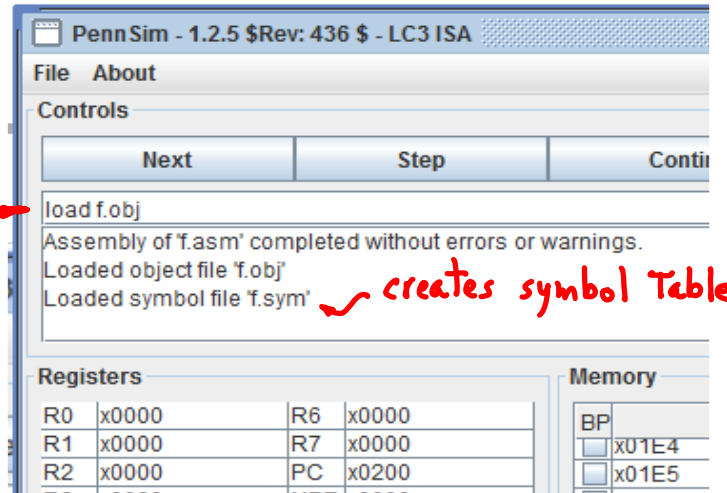
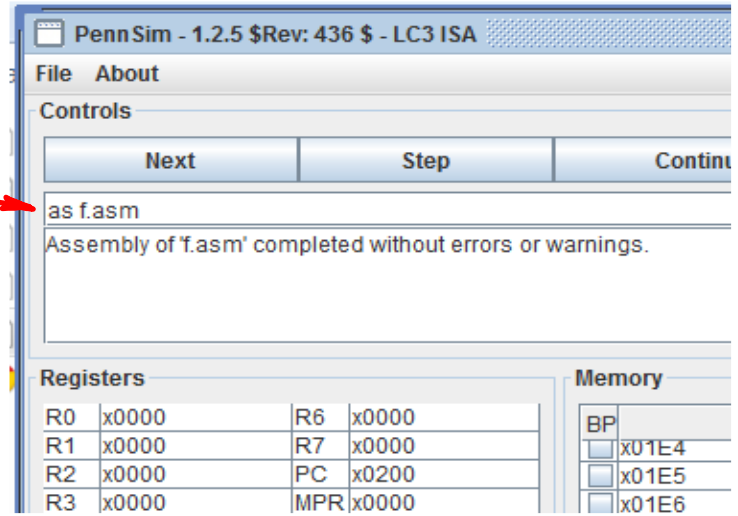
RRRRR == Red intensity (00 - 1F)  
 BBBB == Blue intensity (00 - 1F)  
 GGGG == Green intensity (00 - 1F)

# PennSim Graphics mode

f.asm  
write to VRAM

```
.ORIG x0200
LD R4, PTR
BRnzp MAIN
PTR:
.FILL DATA
MAIN:
LDR R0, R4, #0 ;-- red
LDR R3, R4, #4 ;-- VRAM
STR R0, R3, #0 ;-- (127, 123)
STR R0, R3, #-1 ;-- (127, 122)
STR R0, R3, #-2 ;-- (127, 121)
STR R0, R3, #-3 ;-- (127, 120)
STR R0, R3, #-4 ;-- (127, 119)
STR R0, R3, #-5 ;-- (127, 118)
DATA:
.FILL x7C00 ;-- ORRR RR00
.FILL xC000 ;-- start of VRAM
.FILL xFDFF ;-- end of VRAM
.END
```

assemble f.asm  
(row, col)



load f.obj  
creates symbol Table

Device

<input type="checkbox"/>	x0200	x2801	LD R4, PTR
<input type="checkbox"/>	x0201	x0E01	BR MAIN
<input type="checkbox"/>	x0202 PTR	x020B	BRp x020E
<input type="checkbox"/>	x0203 MAIN	x6100	LDR R0, R4, #0
<input type="checkbox"/>	x0204	x6704	LDR R3, R4, #4
<input type="checkbox"/>	x0205	x70C0	STR R0, R3, #0
<input type="checkbox"/>	x0206	x70FF	STR R0, R3, #-1
<input type="checkbox"/>	x0207	x70FE	STR R0, R3, #-2
<input type="checkbox"/>	x0208	x70FD	STR R0, R3, #-3
<input type="checkbox"/>	x0209	x70FC	STR R0, R3, #-4
<input type="checkbox"/>	x020A	x70FB	STR R0, R3, #-5
<input type="checkbox"/>	x020B DATA	x7C00	STR R6, R0, #0

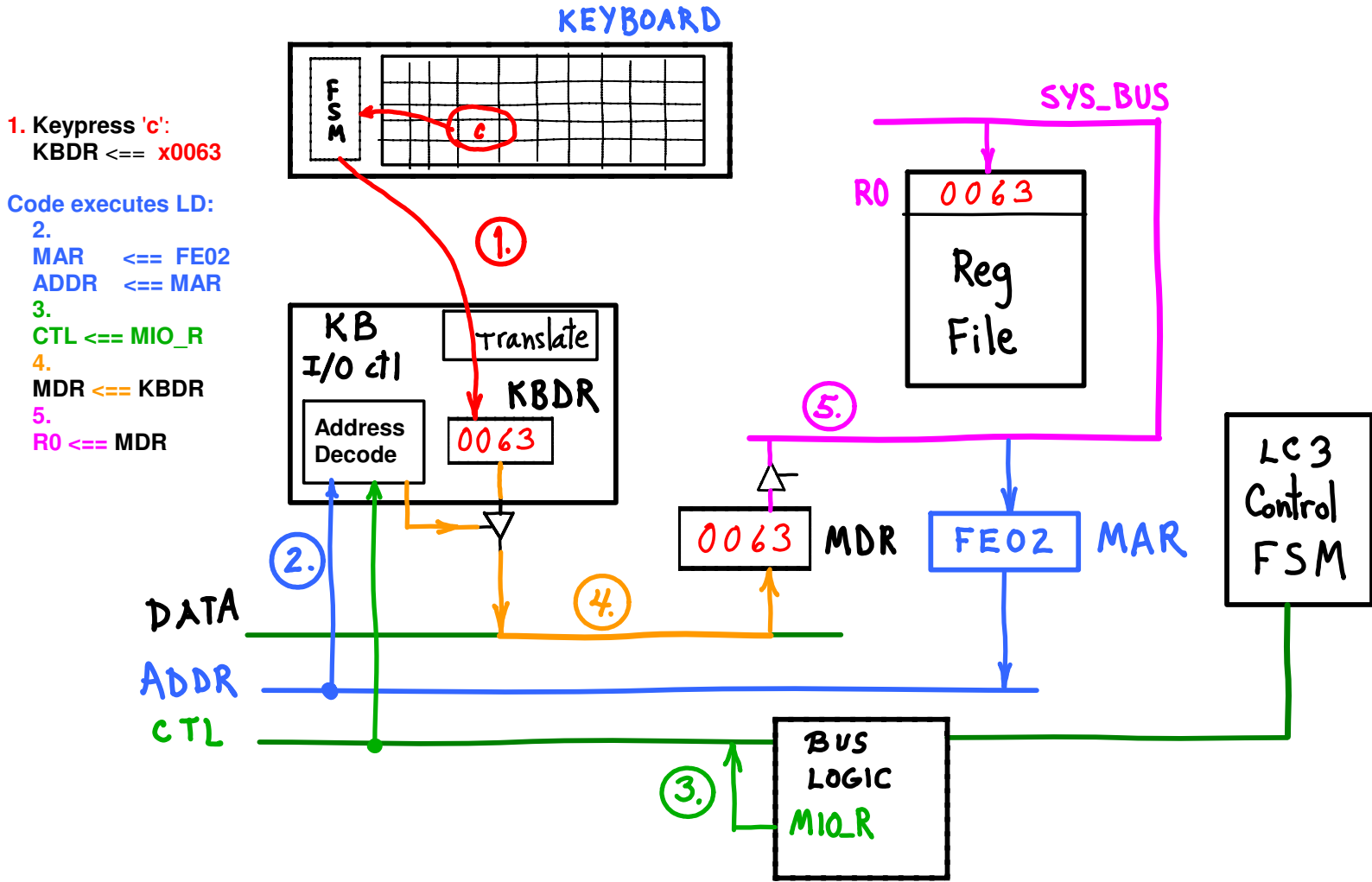
last 6 pixels

f's MAIN

The VRAM is an I/O device. Writing to the range of addresses C000-FDFF sets pixels on the display device.

PennSim shows two displays (text-mode, graphics-mode), but actually there is only one, it can be set in either mode.

# Reading KBDR device register



## Device Polling

DOES KBDR have new data?

==> check KB I/O CTL: status register

POLL:

```
LDI R1, KBSR
BRn Ready
BRnzp POLL
```

Ready:

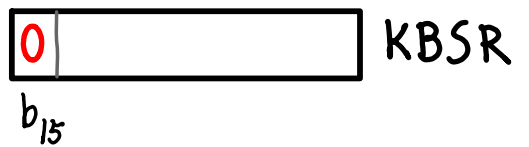
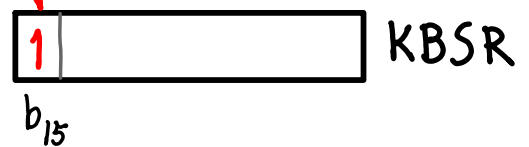
```
LDI R0, KBDR
```

```
KBSR: .FILL xFE00
```

```
KBDR: .FILL xFE02
```

not ready

new data is in KBDR



after KBDR is read,  
KBctl resets b<sub>15</sub>

# LC3-OS I/O Services

TRAP	pseudonym	Description (PP, Append. A.4)
TRAP x25	HALT	jump to OS w/ message, loops in OS forever.
TRAP x20	GETC	one char input, keyboard data ==> R0[7:0] (clears R0 first).
TRAP x21	OUT	one char output, R0[7:0] ==> display; ignores big-end byte, R0[15:8].
TRAP x22	PUTS	string output, Mem[R0++] ==> display until x0000. 1 char per word.
TRAP x23	IN	one char input w/ prompt, ala GETC.
TRAP x24	PUTSP	same as PUTS, but packed (2 chars per word, little-end byte then big-end byte).

## Non-Memory Mapped I/O (Intel x86 IA-32 and IA-64 ISAs)

--- Separate opcodes,

in R4, x0D  
out R4, xC7

--- Separate address spaces,

Control-Bus signal:

0, access memory;  
1, access I/O device (but uses same address wires)

