An FPGA-based Intel 8080 Mockup Soft Microprocessor

Leonardo Cattarin

June 22, 2022

University of Trento

Modules scheme

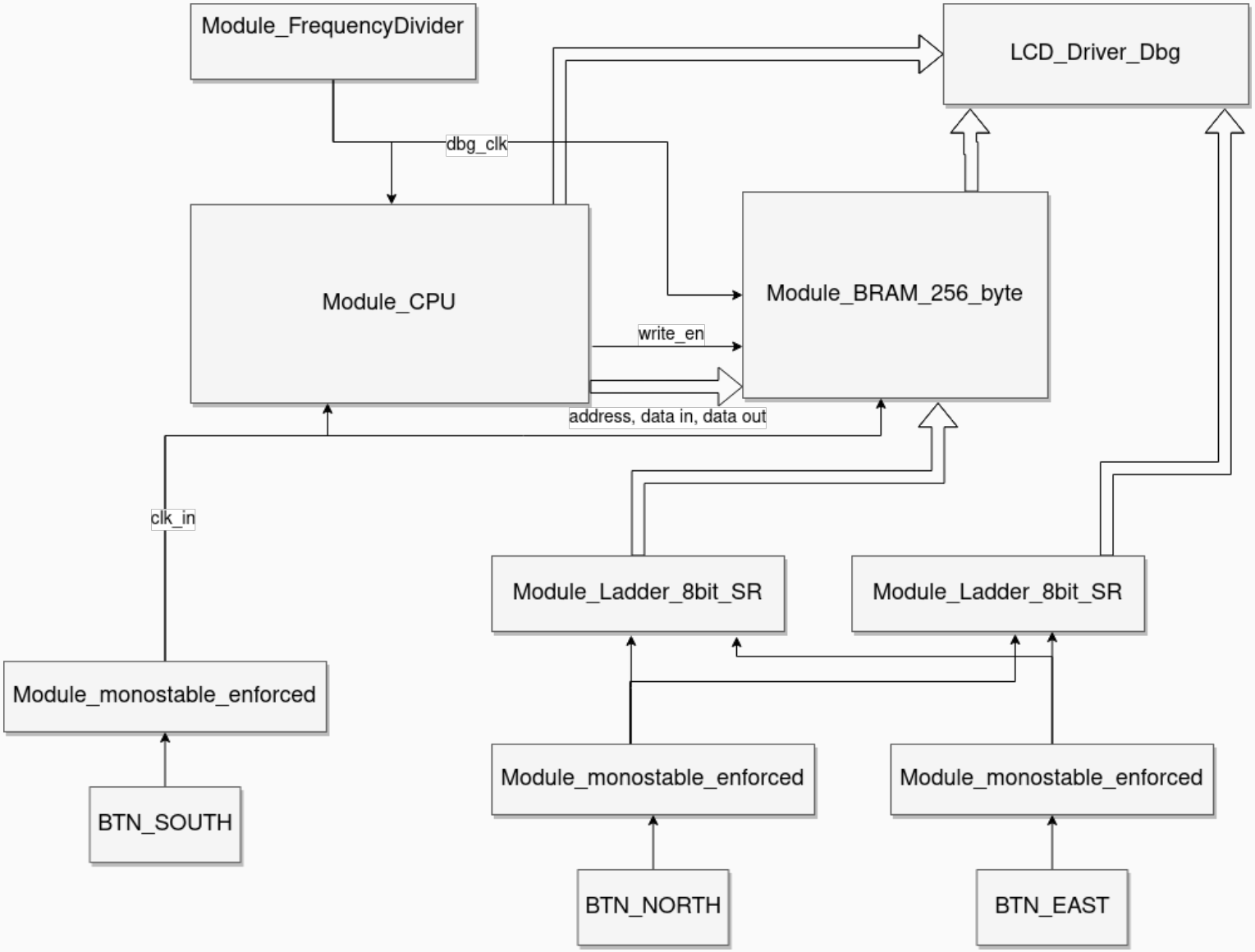

Figure 1: Modules scheme of the computing system

CPU: Instruction Fetch-execute cycle

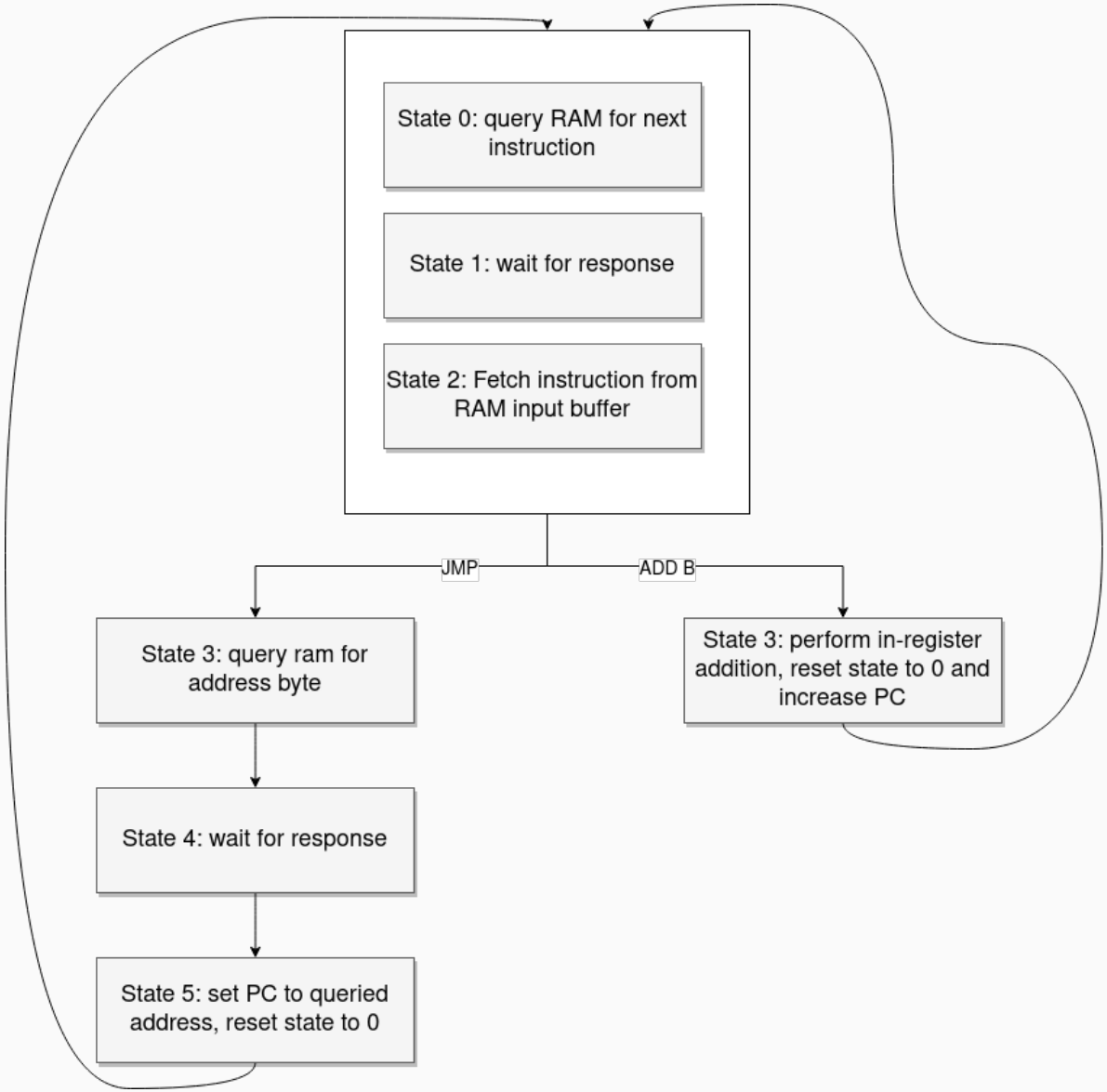

Figure 2: Fetch-execute cycle corresponding to JMP and ADD

CPU: Registers

- `PC` : (8-bit) Program Counter, memory address to the next instruction.
- `IR` : (8-bit) Instruction Register, opcode of current instruction
- `A,B,C` : (8-bit) General purpose registers
- `W,Z` : (8-bit) Temporary registers
- `H,L` : (8-bit) Registers used to store memory addresses
- `data_addr,data_out,write_en` : Registers connected to output wirebuses.
- `flg_carry` , `flg_sign` , `flg_zero` , `flg_parity` , `flg_auxiliary` : flag registers

CPU: instructions

- `NOP` : do nothin
- `JMP XX` : jump to address XX
- `JC XX` : jump if `flg_auxiliary` =1
- `MVI B, XX` : copy immediately byte XX to B
- `MOV B,A` : `MOV B,C` , `MOV C,B` , `MOV B,H` `MOV H,B` : `MOV B,L` , `MOV L,B` , `MOV M(H),B` , `MOV B,M(H)` , copy operations
- `ADD B` : `ADD M[H]` , Add to content to A
- `CMP B` : compare A with B. If A=B, set zero flag to 1. Otherwise set it to zero, and set carry flag to zero if A¿=B or 1 otherwise
- `CHC` : `CHZ` , `CHS` , `CHP` , copy content of flag registers auxiliary one.
- `HLT` : do noting and stop

RAM and LCD Display

`Module_BRAM_256_byte` Module:

- Based on Block RAM functionality of Spartan 3A
- Used as an array of 8-bit registers
- Used only 256 bytes = 8-bit addresses

`LCD_Driver_Dbg` module:

- Configures LCD display and issues a cycle of write commands
- Can show either RAM memory or CPU registers contens
- selectable via switch and pushbuttons

Exampe: For loop

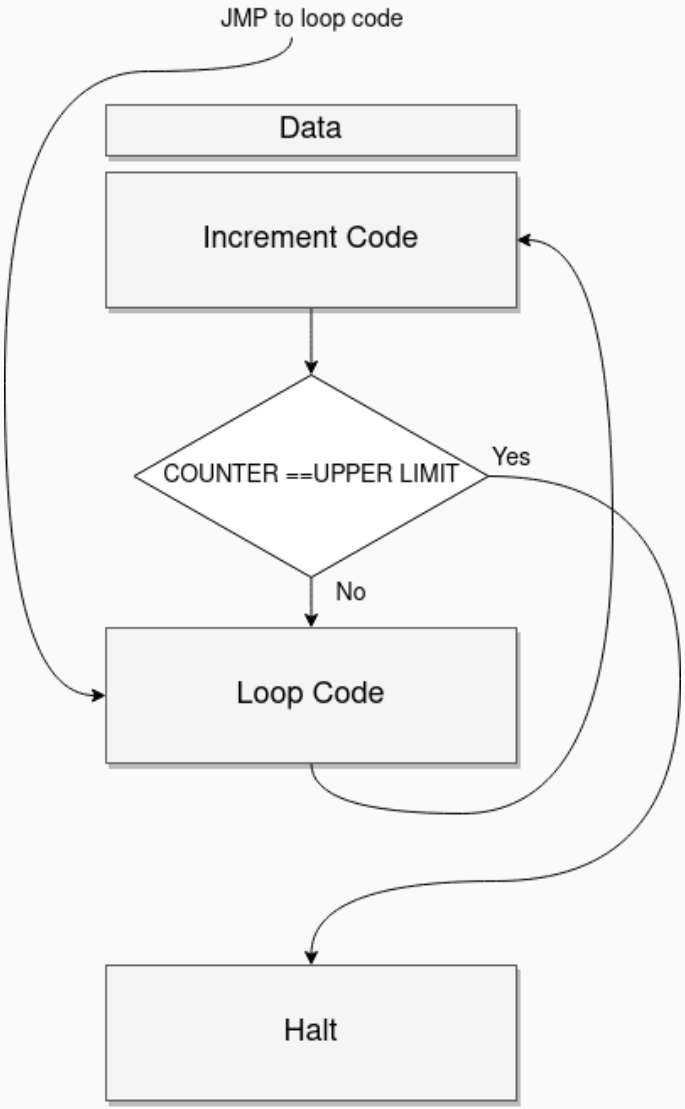

Figure 3: Loop Example Scheme

Exampe: For loop

```
//jump directly to loop instructions
JMP PTR[LOOP]

//initial counter value, upper limit
//and memory location to write numbers.
COUNTER = 0
UPPER_LIMIT = 0A
NUMBERS_ADDR = 35

//code section for counter increment
//and conditional jump
#INCREMENT#

//load counter address in H
MVI B,PTR[COUNTER]
MOV H,B

//put 01 in B, use it to increase counter by 1,
// put result in B
MVI B, 01
ADD B
MOV B,A

//write counter to its memory position
MOV M[H],B

//load upper limit memory location in B
// and then the limit itself
MVI B,PTR[UPPER_LIMIT]
MOV H,B
MOV B,M[H]

//compare A (counter) with B(upper limit)
//if A=B, zero flag is set to 1
CMP B

//check if zero flag is 1 (A=B)
//and jump to end program if true
CHZ
JC PTR[AFTER_LOOP]

#LOOP#
//puts counter address in H
MVI B,PTR[COUNTER]
MOV H,B

//puts counter in B, A and C
MOV B, M[H]
MOV A,B
MOV C,B

//load starting address for numbers writing
MVI B,PTR[NUMBERS_ADDR]
MOV H,B
MOV B,M[H]

//calculates address to write counter value, put in H
ADD B
MOV B,A
MOV H,B

//re-load counter value in B and A from C
//and write it to memory
MOV B,C
MOV A,B
MOV M[H],B

//jump to increment routine
JMP [INCREMENT]

#AFTER_LOOP#
HLT

#NUMBERS_ADDR#
```

Bibliography

[1] [2] [3]

References i

[1] Intel Corporation.
8080/8085 Assembly language programming manual.

[2] Xilinx, Inc.
Spartan-3A/3AN FPGA Starter Kit Board User Guide.

[3] Xilinx, Inc.
Using Block RAM in Spartan-3 Generation FPGAs.