ECE 449 Project

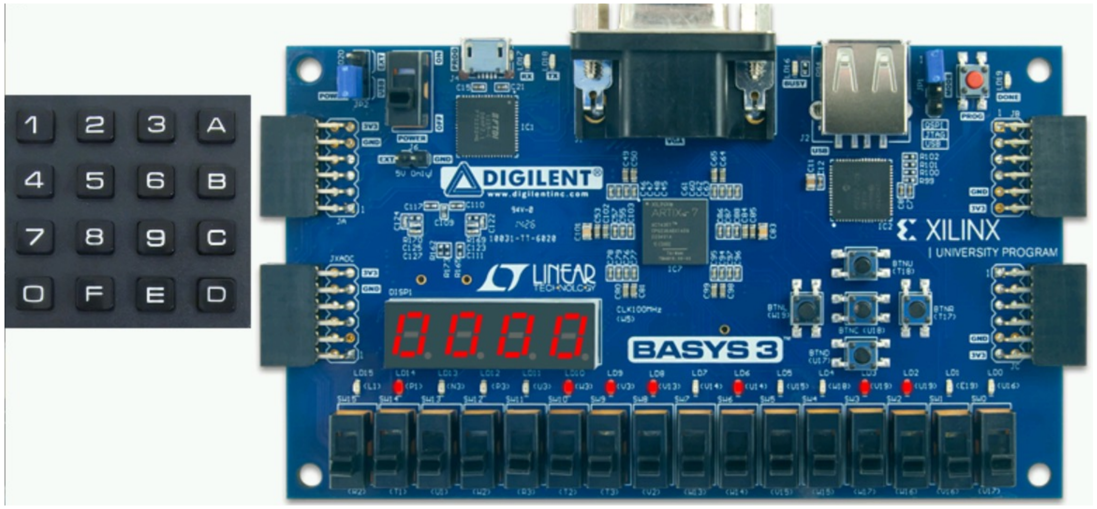
Description

- Design and implement a 16-bit CPU

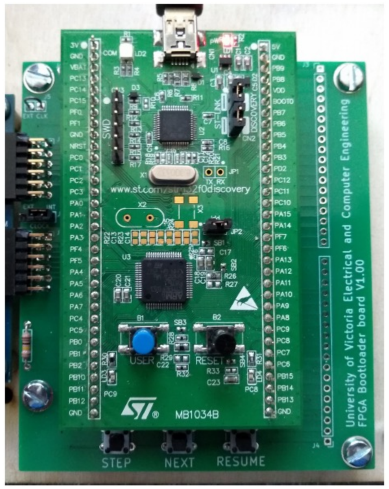
Milestones

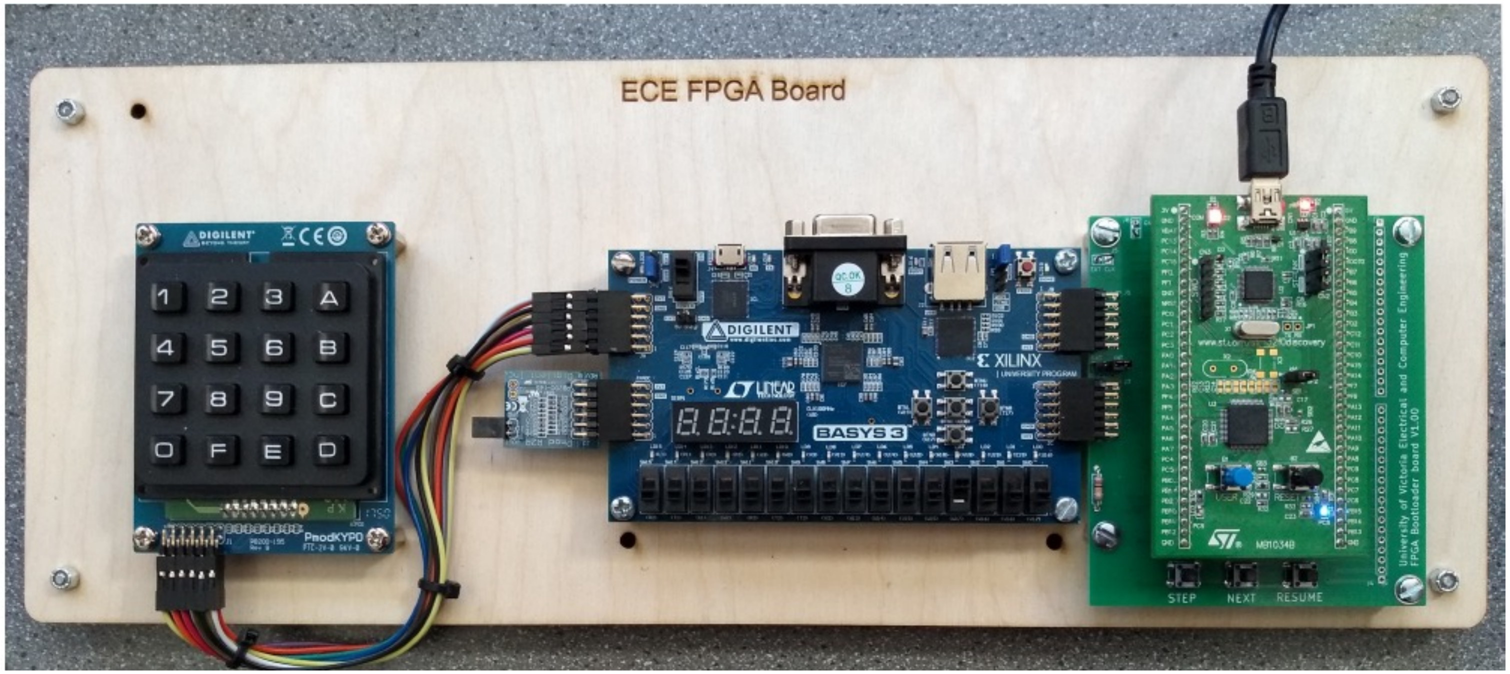
- Preliminary Design Review March 4, 2024
  - Concrete Specifications
  - High Level Block Design
  - Format A instructions
- Format B Instructions March 11, 2024
- Format L Instructions March 25, 2024
- Final Design Review and Project Demo
  (TBD) April 4&8 2024
- Final Project Report (TBD) --- Final exam date

- Basys 3

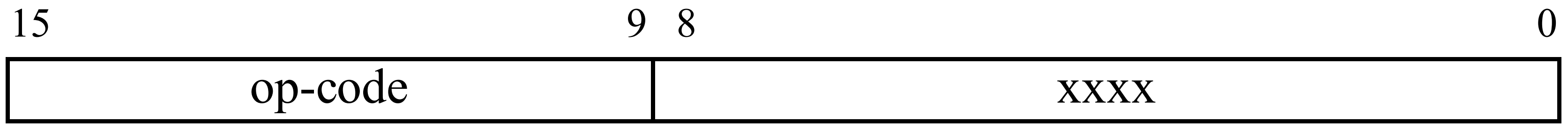

1 2 3 A
4 5 6 B
7 8 9 C
0 F E D
DIGILENT
XILINX
UNIVERSITY PROGRAM
LINEAR TECHNOLOGY
BASYS 3

- FPGA
- STM32F0DISCOVERY Board

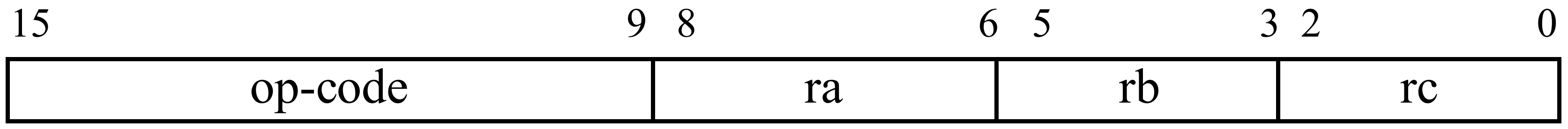

University of Victoria Electrical and Computer Engineering
FPGA Bootloader board V1.00
STEP
NEXT
RESUME
MB1034B
USER
RESET
SWD
www.st.com/stm32f0discovery


ECE FPGA Board
PmodKYPD
BASYS 3
XILINX
DIGILENT
STEP
NEXT
RESUME
University of Victoria Electrical and Computer Engineering
FPGA Bootloader board V1.00

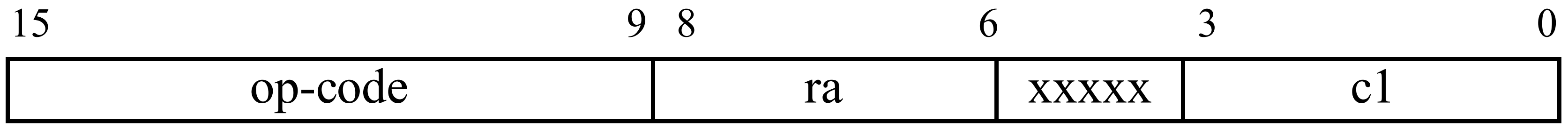
Format A



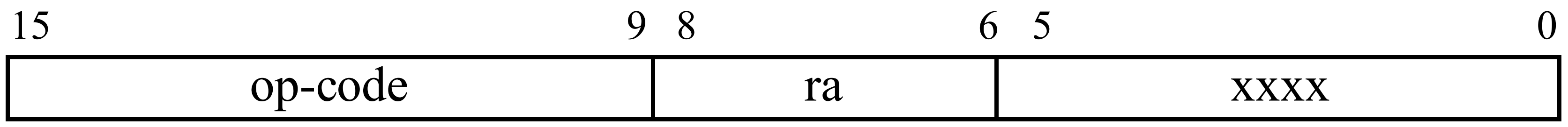
Format A0



Format A1



Format A2



Format A3

16-bit (A Format)

Mne-monic	Op-code	Function	Type	Syntax
NOP	0	Nothing	A0	*NOP*
ADD	1	R[ra] ← R[rb] + R[rc];	A1	*ADD ra,rb,rc*
SUB	2	R[ra] ← R[rb] – R[rc];	A1	*SUB ra,rb,rc*
MUL	3	R[ra] ← R[rb] × R[rc];	A1	*MUL ra,rb,rc*
NAND	4	R[ra] ← R[ra] NAND R[rb]:	A1	*NAND ra,rb,rc*

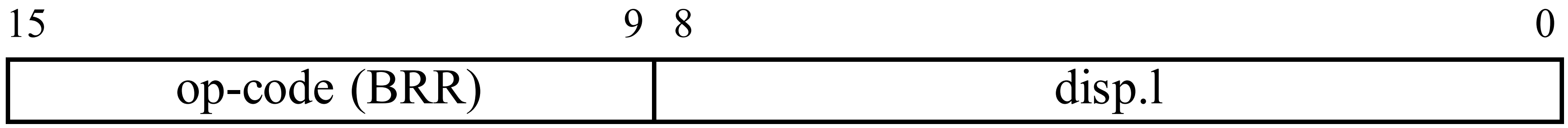
SHL	5	n:=c1<3…0>):
		(n|0) → (R[ra]<15…0>←R[ra]<15-n…0>#(n@0)):
		(n=0) → () ;
			A2	*SHL ra#n*

SHR	6	n:=c1<3…0>):
		(n|0) → (R[ra]<15…0>←(n@0)#R[ra]<15…n>):
		(n=0) → () ;
			A2	*SHR ra#n*

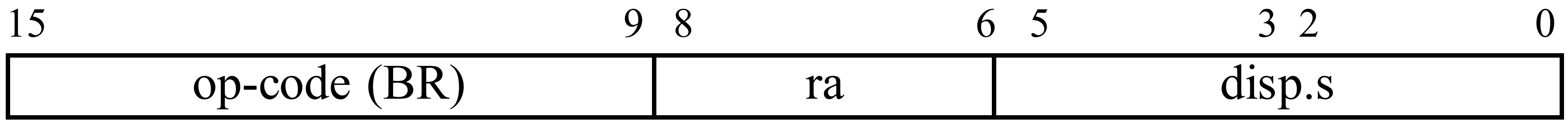
TEST	7	(R[ra) = 0) →Z ← 1; else →Z ← 0: (R[ra] < 0) →N ← 1; else →N ← 0;	A3	*TEST ra*
OUT	32	OUT.PORT ← R[ra];	A3	*OUT ra*
IN	33	R[ra] ← IN.PORT;	A3	*IN ra*

Multiplication needs to be refined. This will affect the TEST and the flags

Format B (Branch)



Format B1



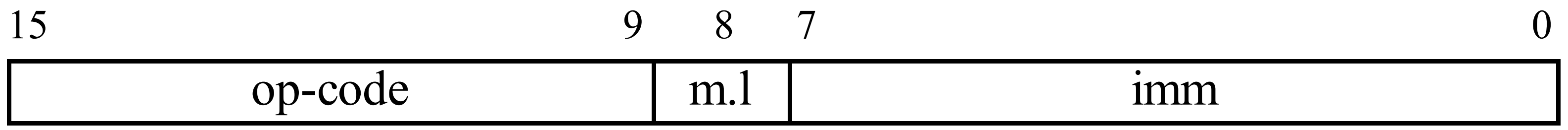
Format B2

16-bit (B Format)

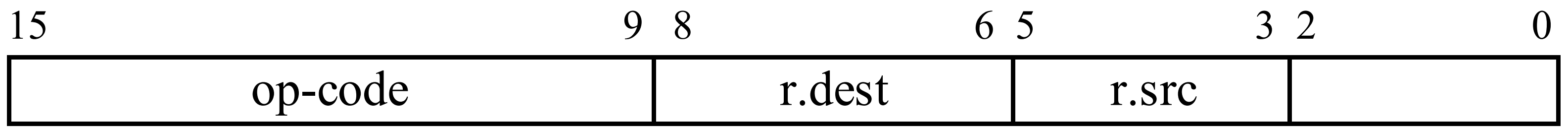
Mne-monic	Op-code	Function	Type	Syntax
BRR	64	PC ← PC+2*disp.l {sign extended 2's complement}	B1	*BRR +disp.l*
BRR.N	65	(N=1) → PC ← PC+2*disp.l {sign extended 2's complement}; (N=0) → PC ← PC+2 { 2's complement};	B1	*BRR.N +disp.l*
BRR.Z	66	(Z=1) → PC ← PC+2*disp.l {sign extended 2's complement}; (Z=0) → PC ← PC+2 { 2's complement};	B1	*BRR.Z +disp.l*
BR	67	PC ← R[ra]+2*disp.s {sign extended 2's complement}	B2	*BR ra+disp.s*
BR.N	68	(N=1) → PC ← R[ra] {word aligned}+2*disp.s {sign extended 2's complement}; (N=0) → PC ← PC+2 { 2's complement};	B2	*BR.N ra+disp.s*
BR.Z	69	(Z=1) → PC ← R[ra] {word aligned}+2*disp.s {sign extended 2's complement}; (Z=0) → PC ← PC+2 { 2's complement};	B2	*BR.Z ra+disp.s*
BR.SUB	70	r7 ← PC + 2; PC ← R[ra] {word aligned}+2*disp.s {sign extended 2's complement};	B2	*BR.SUB ra+disp.s*
RETURN	71	PC ← r7;	A0	*RETURN*

For the PC operations, the argument is the value of the PC just before the instruction is fetched, while the result is the value of the PC at the conclusion of the execution of the instruction.

Formal L (load/store)



Format L.1



Format L.2

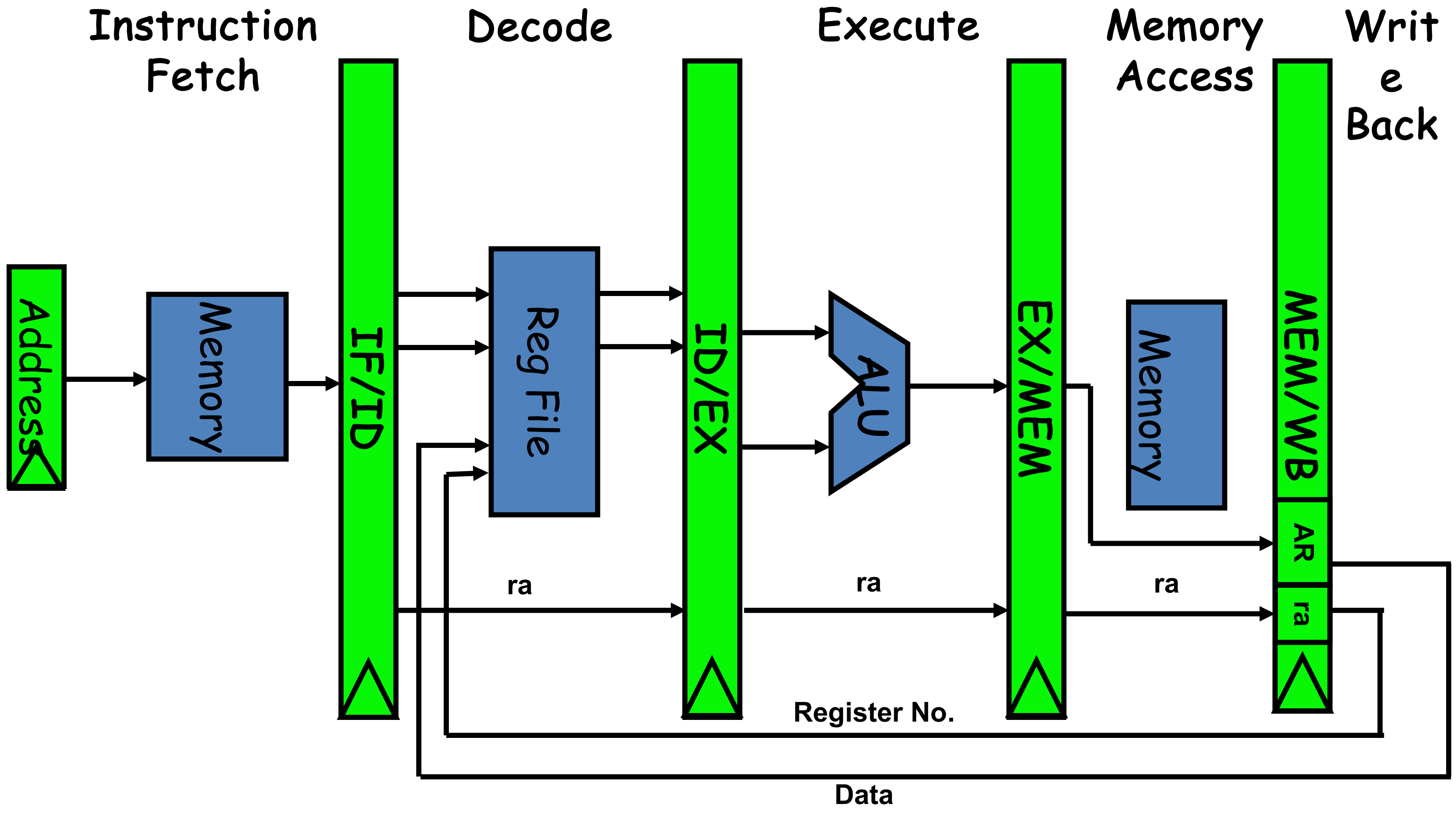
16-bit (L Format)

Mnemonic	Op-code	Function	Type	Syntax
LOAD	16	R[r.dest] ← M[R[r.src]];	L2	*LOAD r.dest, r.src* *LOAD r.drst, @r.src*
STORE	17	M[R[r.dest]] ← R[r.src];	L2	*STORE r.dest, r.src* *STORE @r.dest, r.src*
LOADIMM	18	(m.l=1) →R7⟨15⋯8⟩← imm: (m.l=0) →R7⟨7⋯0⟩← imm;	L1	*LOADIMM.upper #n* *LOADIMM.lower #n*
MOV	19	R[r.dest] ←R[r.src];	L2	*MOV dest,src*

16-bit (Optional)

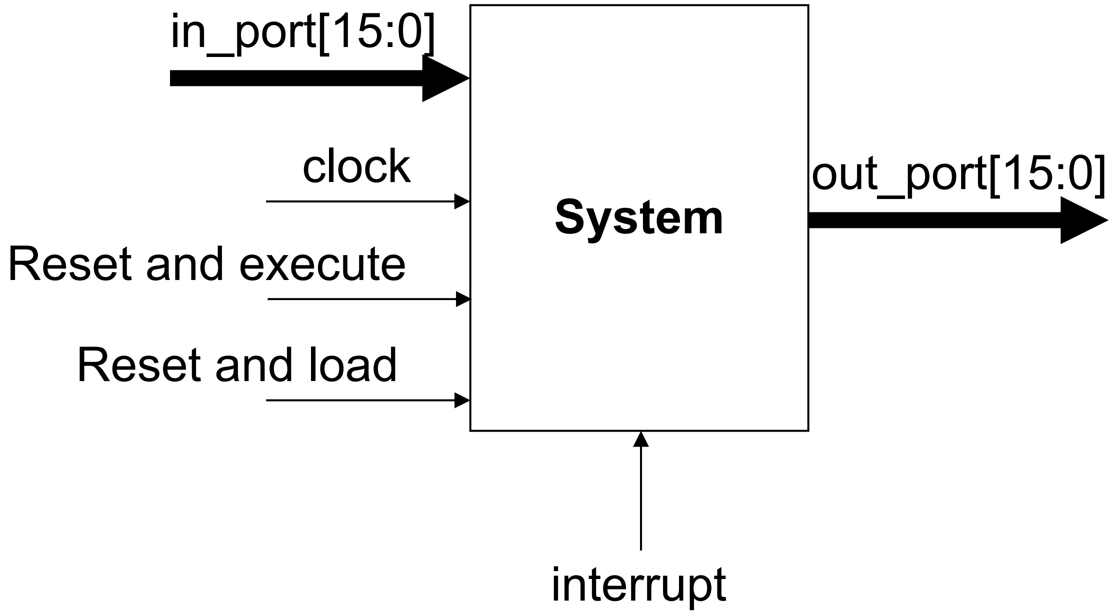
Mne- monic	Op- code	Function		
PUSH	96	M[SP--] ← R[ra];	A3	*PUSH ra*
POP	97	R[rb] ← M[++SP];	A3	*POP ra*
LOAD.SP	98	SP←R[ra];	A3	*LOAD.SP ra*
RTI	99	PC ← M[++SP]: {Z,N} restored	A0	*RTI*

Write Back


Instruction Fetch
Decode
Execute
Memory Access
Writ e Back
Address
Memory
IF/ID
Reg File
ID/EX
ALU
EX/MEM
Memory
MEM/WB
AR
ra
ra
ra
ra
Register No.
Data

Pinout of Processor

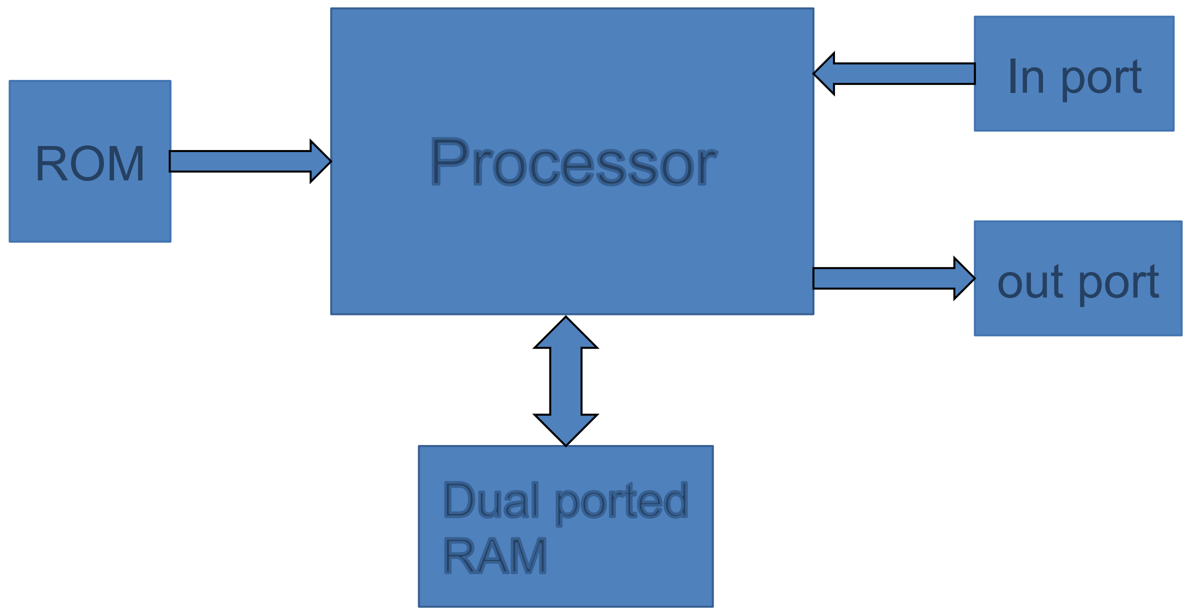
- Interrupt is optional



Resets

- Reset and execute causes the system to execute the user’s code
- Reset and load causes the system to load the user’s code into RAM
- Explanations and detailed specifications will follow in this slide

System Description



Comments on the system architecture -RAM

- A dual ported RAM is used to ensure that instruction and data traffic is separated (Harvard architecture)
- The RAM serializes the access requests arbitrarily.
- The suggested (slide 22) RAM implementation is a synchronous one and it allows the specification of the delay (in clock cycles) of the read data to appear on the data_out port of the RAM.

Comments on the system architecture -ROM

- ROM is used to store a rudimental BIOS.
- ROM and its BIOS will be provided to you
- The main functionality of the BIOS is
  - Load user code into the appropriate location in RAM
  - Execute user code

Resets

- The two resets implement the load and execute functionality of the BIOS
- Both clear the PC
- **Reset and Execute** *vectors* to address 0x0000 while **Reset and Load** *vectors* to address 0x0002.
- At each address, the developer has introduced the appropriate branch (BRR) instruction that vectors to the reset-handling routine (this is part of the BIOS)

ROM, RAM and ports

- ROM is 1024-byte large starting at address 0x0000
- RAM is a 1024 byte block starting at address 0x0400
- We use memory-mapped ports. The input port is located at 0xFFF0 while the output port is at 0xFFF2

RAM module

- Please use the dual port distributed RAM macro XPM_MEMORY_DPDISTRAM from Xilinx XPM macro group
- This macro can configure a dual ported memory where port A can be used for both reading and writing while port B can only be used for reading only (from memory).

Comments

- Your Lab demonstrator will elaborate in lab.
- The lab website has the instruction set specs
- We shall discuss main pipelining issues in the next few weeks.