

Single Cycle Processor Instructions

Last updated 4/14/21

Single Cycle Processor - Instructions

- Functionality
 - Harvard Architecture
 - RISC – Load/Store Instruction Set
 - 16 bit instruction words
 - 4 – 8 bit data registers available for executing instructions (A-B-C-D)
 - Support for 16 instructions
 - 3 – memory based instructions

Single Cycle Processor - Instructions

- Instruction Fetch

- Clock the Program Counter (PC)

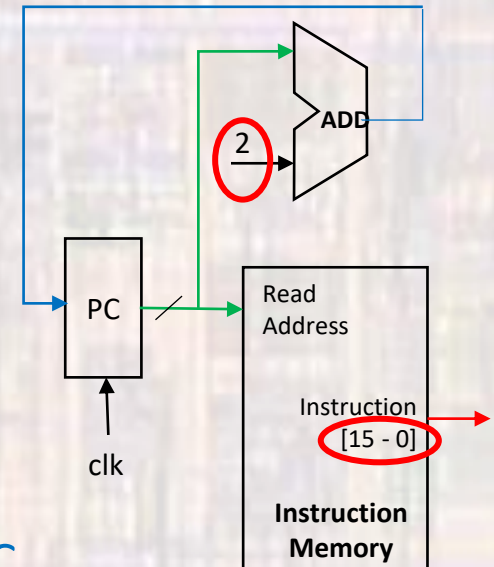
- Register holding the memory address for the NEXT instruction to fetch

- New address is provided to the memory
- Memory provides instruction to its output

- Next address is provided to the input to the PC

- Memory is Byte Addressed
- Instructions are 2 bytes wide
- → increment by 2

- Adder is drawn as an ALU but actual implementation would be our optimized adder block



Single Cycle Processor - Instructions

- Instruction Fetch

- Implementation comments

- We will replace the PC with a sequencer (FSM)

- Sequencer acts as the address register in the memory block

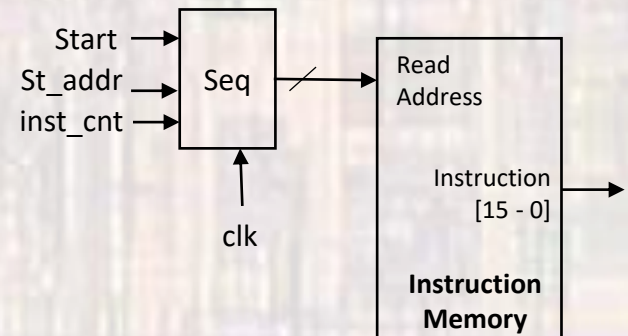
- **Array based** ROM with constants

- Only way to get an asynchronous version

- 4Kb in a x16 configuration

→ asynchronous address

→ asynchronous read



Single Cycle Processor - Instructions

- Arithmetic Instruction Format

Instruction source-1, source-2, destination
 where source-1, source-2, and destination are registers A-D

fn Reg1, Reg2, Wreg

Wreg \leftarrow Reg1 fn Reg2

A	0x12
B	0x23
C	0x35
D	0xF5

A	0x12
B	0xF7
C	0x35
D	0xF5

A	0x12
B	0xF7
C	0x35
D	0x23

- Examples

- add RA, RB, RC

RC \leftarrow RA + RB

- or RB, RD, RB

RB \leftarrow RB or RD

- sub RC, RA, RD

RD \leftarrow RC - RA

Single Cycle Processor - Instructions

- Memory Instruction Format

Instruction address_register, destination_register

`ld` Reg1, Wreg Wreg \leftarrow MEM(Reg1)

Instruction address_register, source_register

`st` Reg1, Reg2 MEM(Reg1) \leftarrow Reg2

- Examples

- `ld` RA, RC

RC \leftarrow MEM(RA)

- `st` RD, RB

MEM(RD) \leftarrow RB

A	0x02
B	0x23
C	0x44
D	0xFE

0x00	0xC3
0x01	0x85
0x02	0x44
...	
0xFE	0x23
0xFF	0x2B

Single Cycle Processor - Instructions

- Load Immediate Instruction Format

Instruction destination_register, value

`ldi Wreg, "imm value"` $Wreg \leftarrow \text{"imm value"}$

- Examples

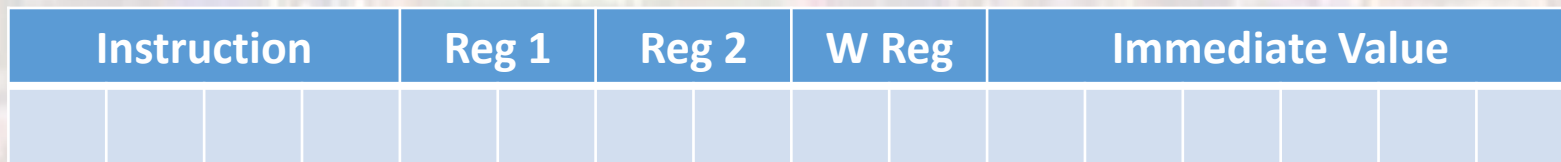
- `ldi RA, 0x12`

$RA \leftarrow 0x12$

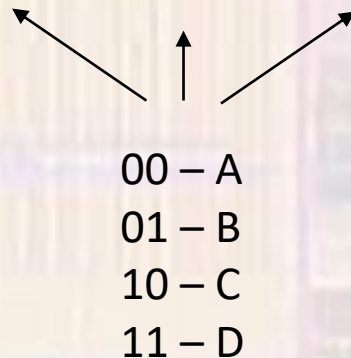
A	0x12
B	0x23
C	0x44
D	0xFE

Single Cycle Processor - Instructions

- Instruction Encoding
 - 16 bits



or	↑	0000
and		0001
nor		0010
nand		0011
add		0100
sub		0101
slt		0110
ld		1000
st		1001
ldi		1100



00 – A
01 – B
10 – C
11 – D

signed Hex
0x20 to 0x1F

100000 to 011111
-32 to 31

Single Cycle Processor - Instructions

- Instruction Encoding – reg/reg

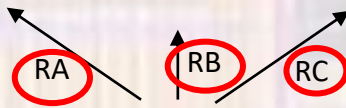
- **sub**
- **sub** RA, RB, RC

Wreg \leftarrow Reg1 - Reg2

RC \leftarrow RA - RB

Instruction				Reg 1		Reg 2		W Reg		Immediate Value					
0	1	0	1	0	0	0	1	1	0	0	0	0	0	0	0

- or 0000
- and 0001
- nor 0010
- nand 0011
- add 0100
- sub 0101**
- slt 0110
- ld 1000
- st 1001
- ldi 1100



- 00 – A
- 01 – B
- 10 – C
- 11 – D

Technically these are don't care but we will always code them as 0s

Single Cycle Processor - Instructions

- Instruction Encoding – Mem

- **ld**
- **ld** RB, RC

$Wreg \leftarrow MEM(Reg1)$

$RC \leftarrow MEM(RB)$

Instruction				Reg 1		Reg 2		W Reg		Immediate Value					
1	0	0	0	0	1	x	x	1	0	0	0	0	0	0	0

or	↑	0000	
and		0001	
nor		0010	
nand		0011	
add		0100	00 – A
sub		0101	01 – B
slt		0110	10 – C
ld		1000	11 – D
st		1001	
ldi		1100	



Technically these are don't care but we will always code them as 0s

Single Cycle Processor - Instructions

- Instruction Encoding – Mem

- st

MEM(Reg1) ← Reg2

- st RB, RC

MEM(RB) ← RC

Instruction				Reg 1		Reg 2		W Reg		Immediate Value					
1	0	0	1	0	1	1	0	x	x	0	0	0	0	0	0

or	↑	0000
and	↑	0001
nor		0010
nand		0011
add		0100
sub		0101
slt		0110
ld		1000
st		1001
ldi		1100



00 – A
 01 – B
 10 – C
 11 – D

↑
 Technically these are
 don't care
 but

we will always code them as 0s

Single Cycle Processor - Instructions

- Instruction Encoding – Immediate

- **ldi**

Wreg \leftarrow “imm value”

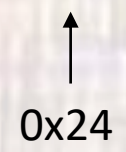
- **ldi** RD, 0x24

RD \leftarrow 0x24

Instruction				Reg 1		Reg 2		W Reg		Immediate Value					
1	1	0	0	x	x	x	x	1	1	1	0	0	1	0	0

- or 0000
- and 0001
- nor 0010
- nand 0011
- add 0100
- sub 0101
- slt 0110
- ld 1000
- st 1001
- ldi 1100**

- 00 – A
- 01 – B
- 10 – C
- 11 – D



Single Cycle Processor - Instructions

- Special function

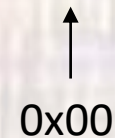
- nop
- nop

do nothing

Instruction				Reg 1		Reg 2		W Reg		Immediate Value					
1	1	1	1	x	x	x	x	x	x	0	0	0	0	0	0



- 00 – A
- 01 – B
- 10 – C
- 11 – D



Single Cycle Processor - Instructions

- Instruction Format

Instruction				Reg 1		Reg 2		W Reg		Immediate Value				

or	0000
and	0001
nor	0010
nand	0011
add	0100
sub	0101
slt	0110
ld	1000
st	1001
ldi	1100
nop	1111

