Instructions

Philipp Koehn

11 September 2019

HW1 – due Friday 9/13
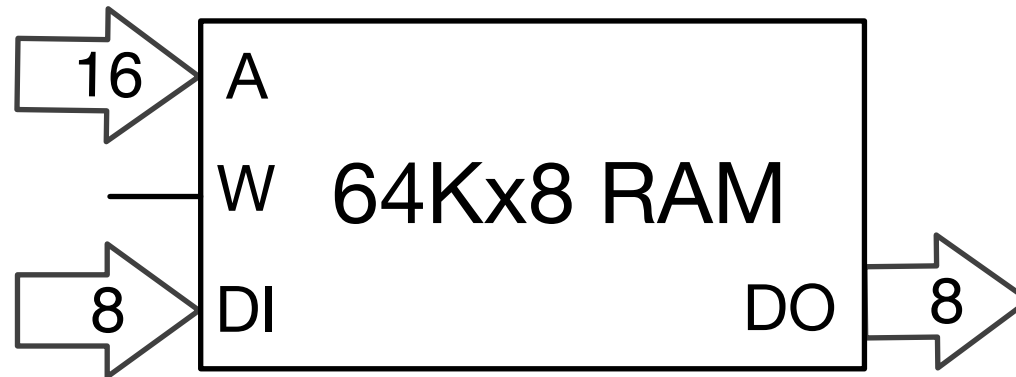
HW2 – due Friday 9/20

⇓

digital design

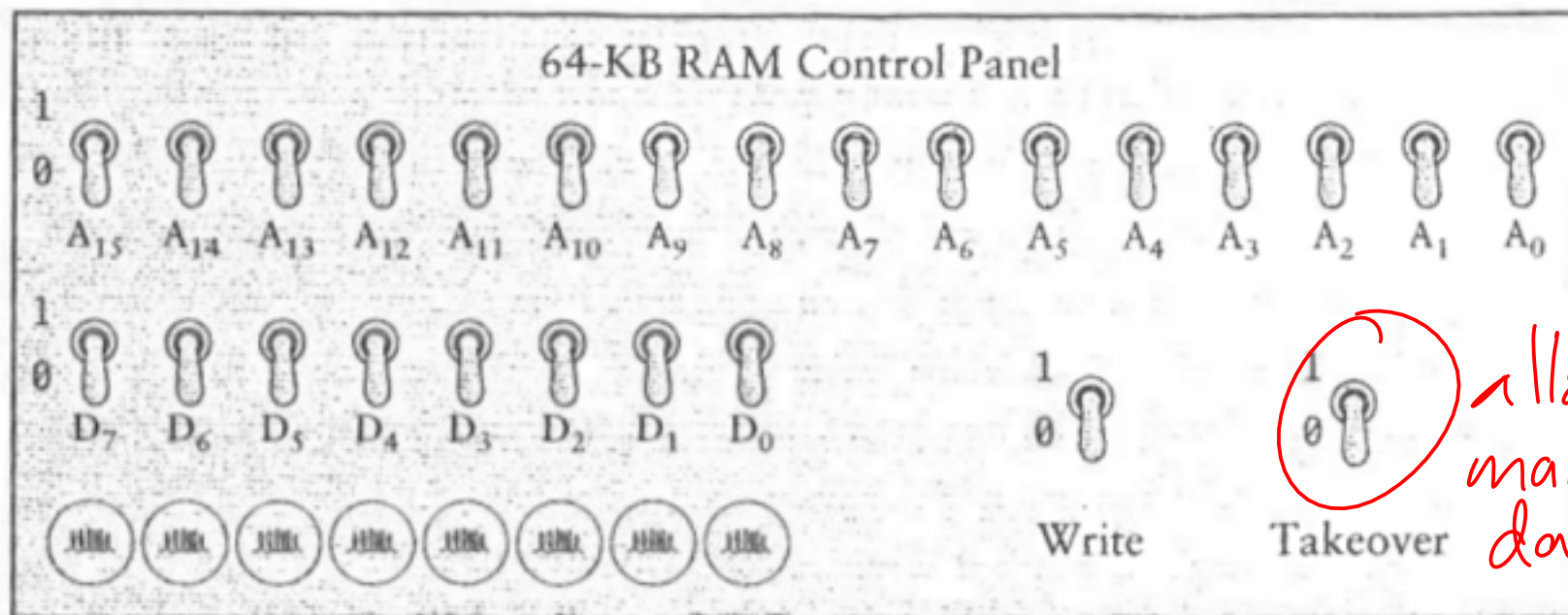# number adder

Design Goal

- Build a machine that adds several numbers together

- Numbers stored in 64 KB RAM

- Idea: Loop through memory with ripple counter

64 KB RAM

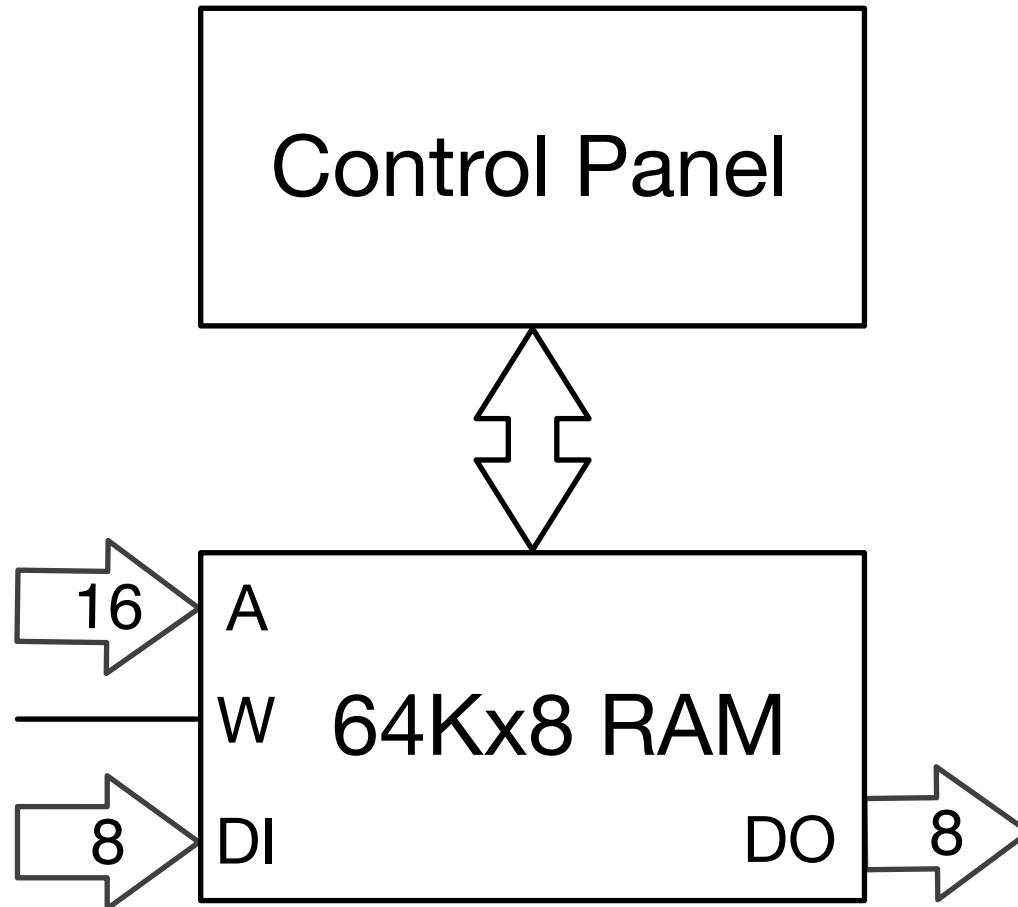

- Read/write 8 bits at a time (one byte)

- 16 bit address space: 2^{16}=65,536 bytes

64-KB RAM Control Panel

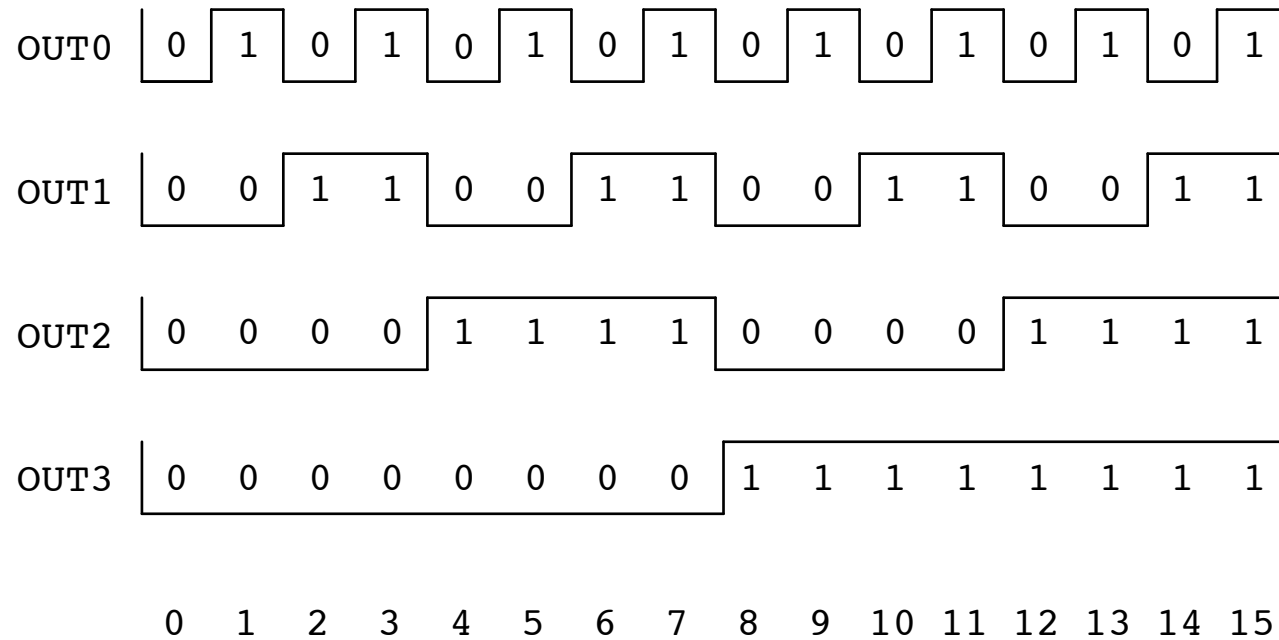allow manual data entry

Control Panel

We can enter numbers and inspect with a control panel

Ripple Counter
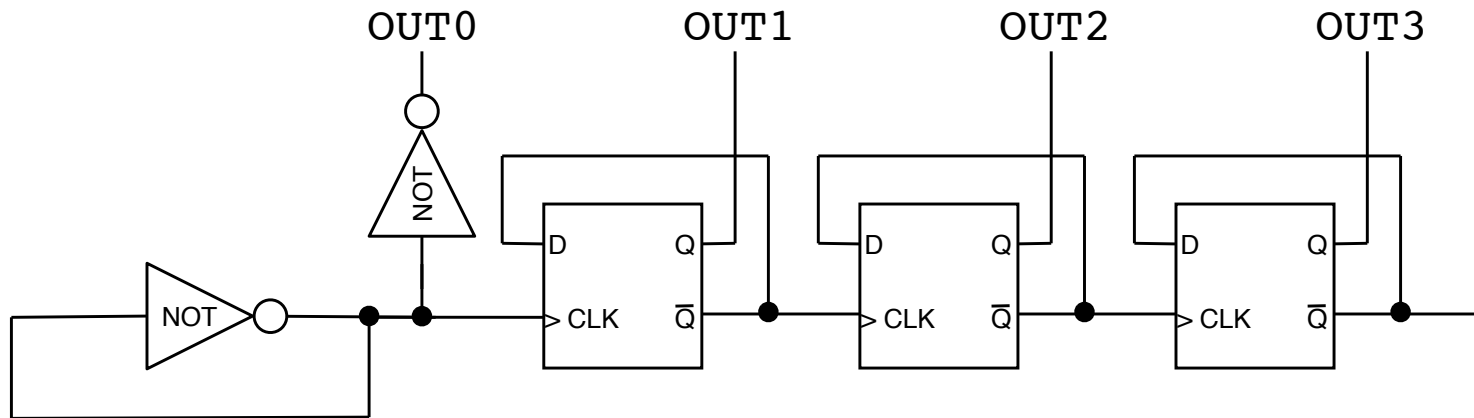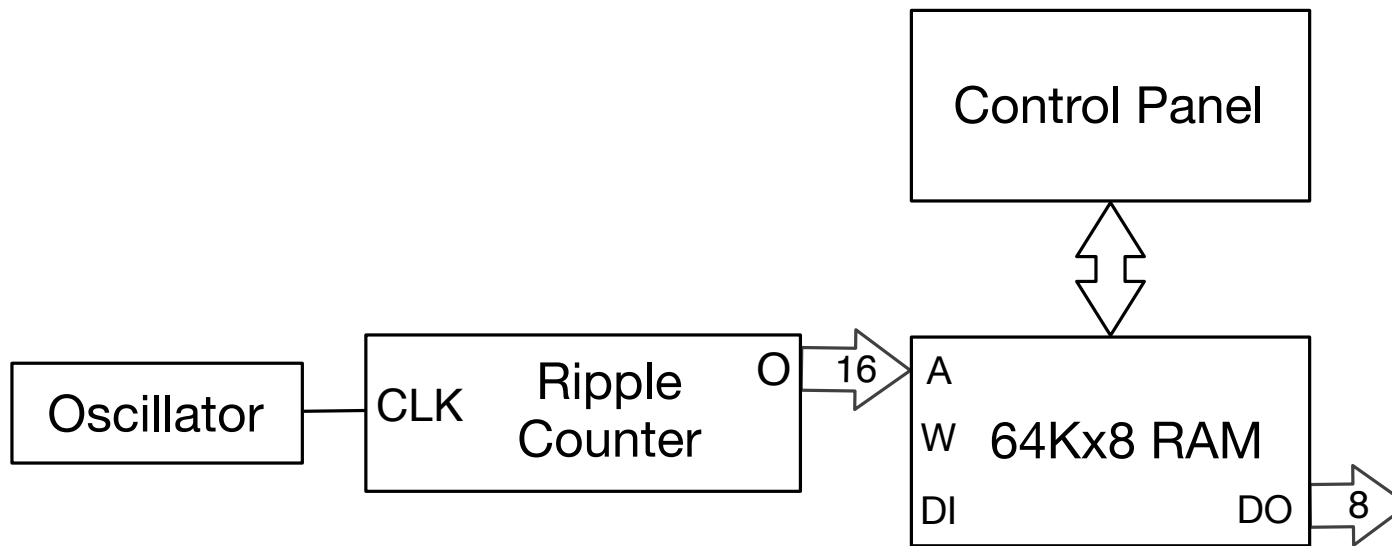
- Ripple counter rotates through number 0, 1, ...

- Each clock cycle, a new number is emitted from memory

Adder
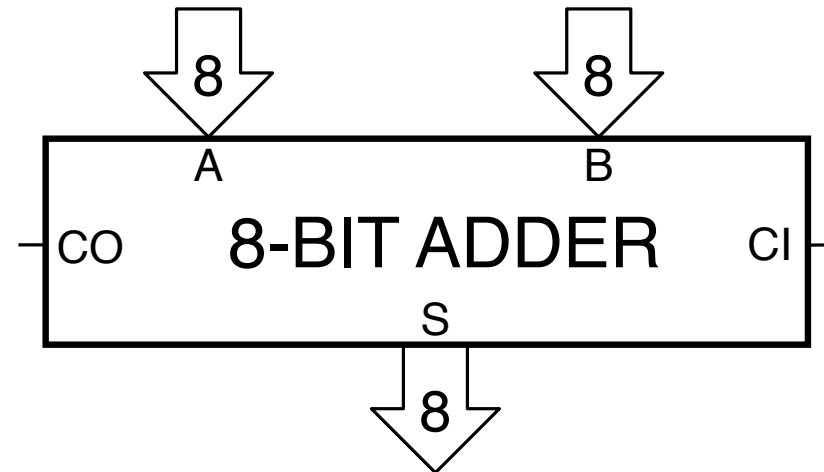
- Adds two numbers: S=A+B

- Overflow: Carry out (CO)

Latch
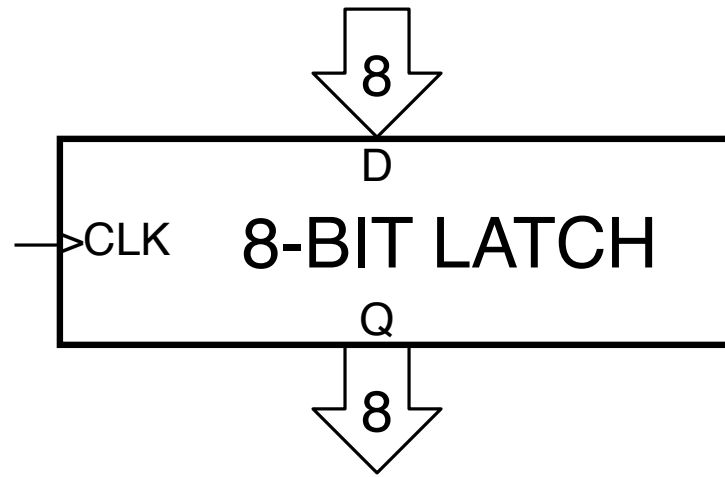
- 8-bit memory

- Edge-triggered: stores value when clock turns to 1

- To be used as *accumulator*

Connecting Adder and Latch

DATA

8

8

A   B

8-BIT ADDER

CO   S   CI

8

CLOCK →CLK   D

8-BIT LATCH

Q

8

OUT
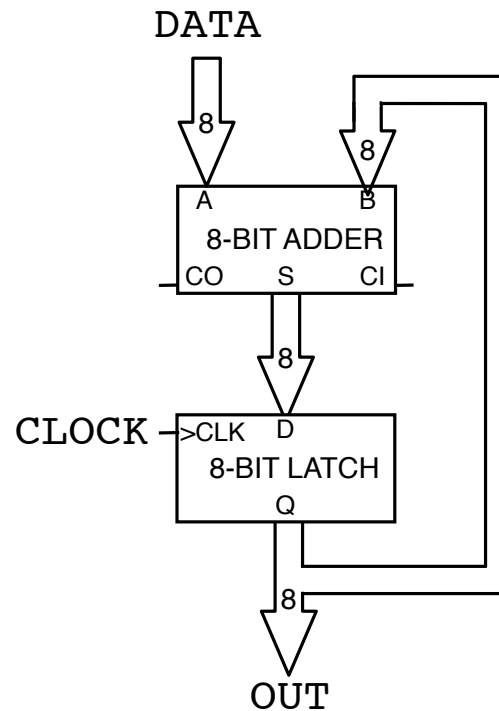
- Adder adds new value (DATA) to accumulator

- Edge trigger prevents immediate feedback

- Output (OUT) may be shown with light bulbs

Number Adder
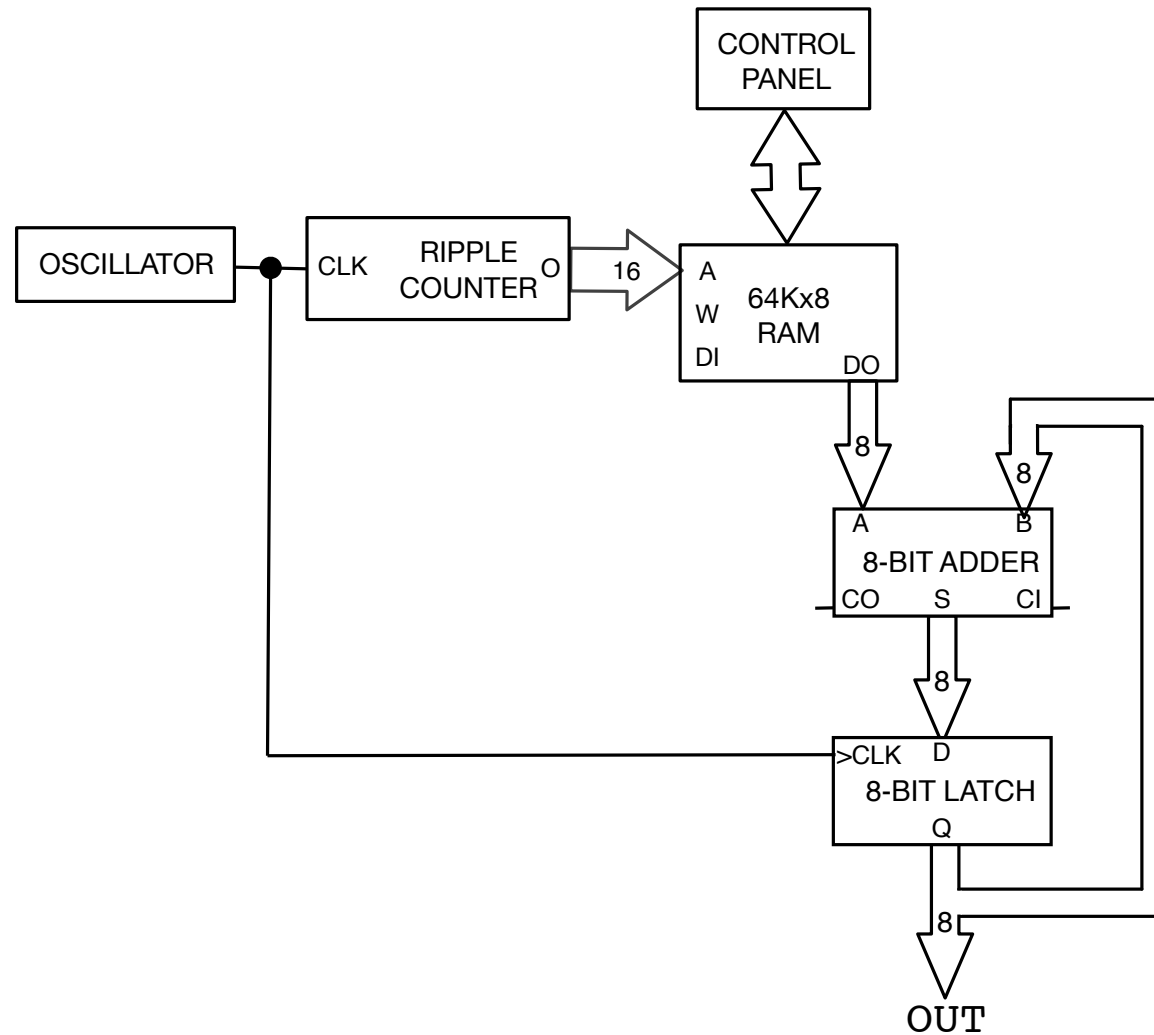
- Everything
  put together

Halt
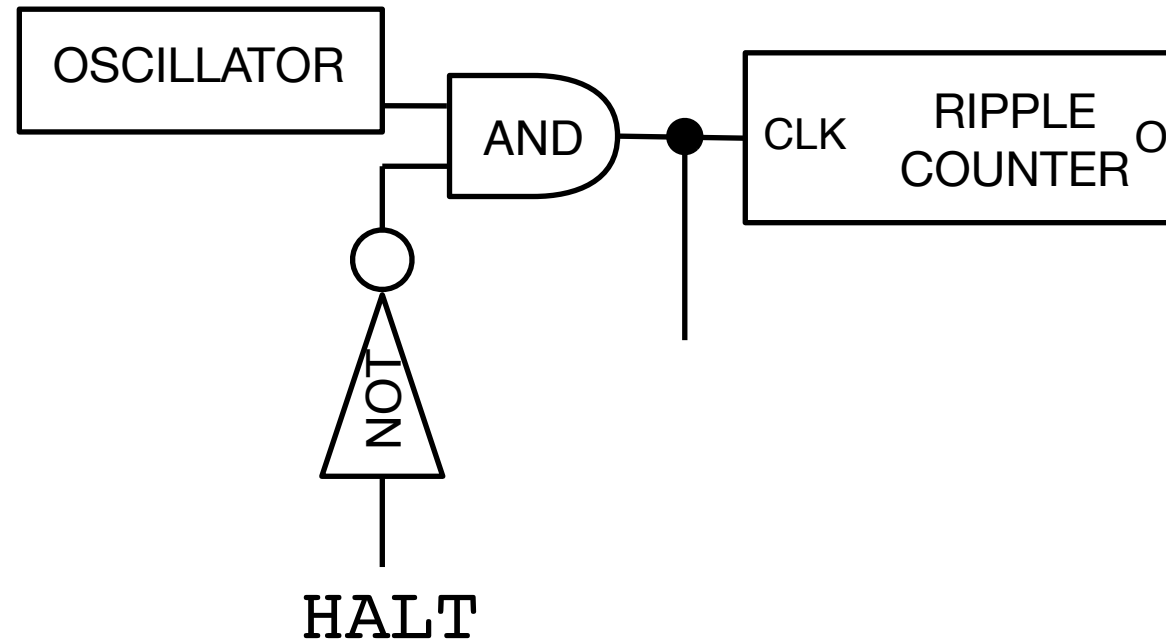
- Halt when external switch is turned on

- Or: cut connection to clock if ripple counter reaches final number

multiple operations

Modification: Alternate Add and Subtract

- Let's say we want to alternate between adding and subtracting

- We already built an integrated adder and subtractor
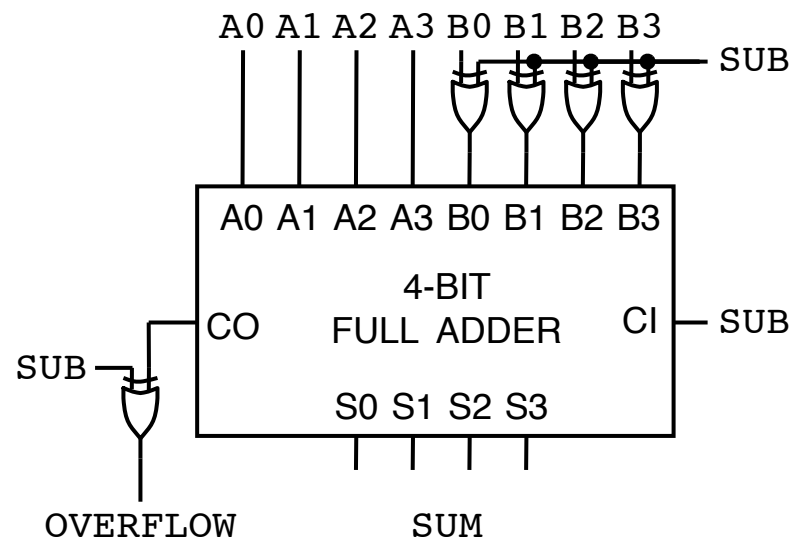
- Idea: indicate operation from last bit of ripple counter

Alternate Add and Subtract
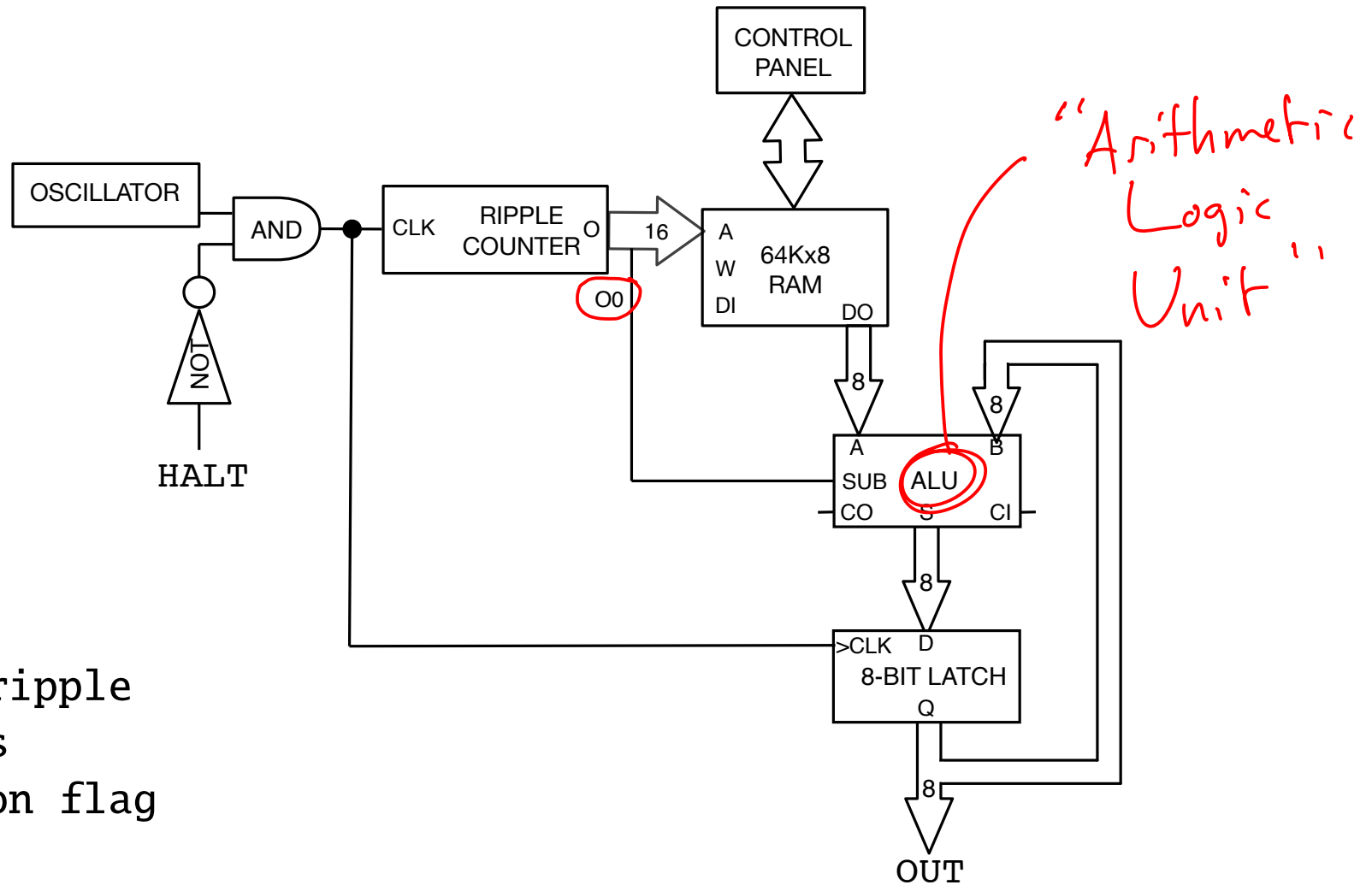
- Bit 0 of ripple counter as instruction flag

instructions

- Control operations by instructions stored in memory

$\Rightarrow$ A *programmable* computer

- Control operations by instructions stored in memory

$\Rightarrow$ A *programmable* computer

- First idea

  – separate instruction memory
  – instructions: add or subtract

Instruction Memory
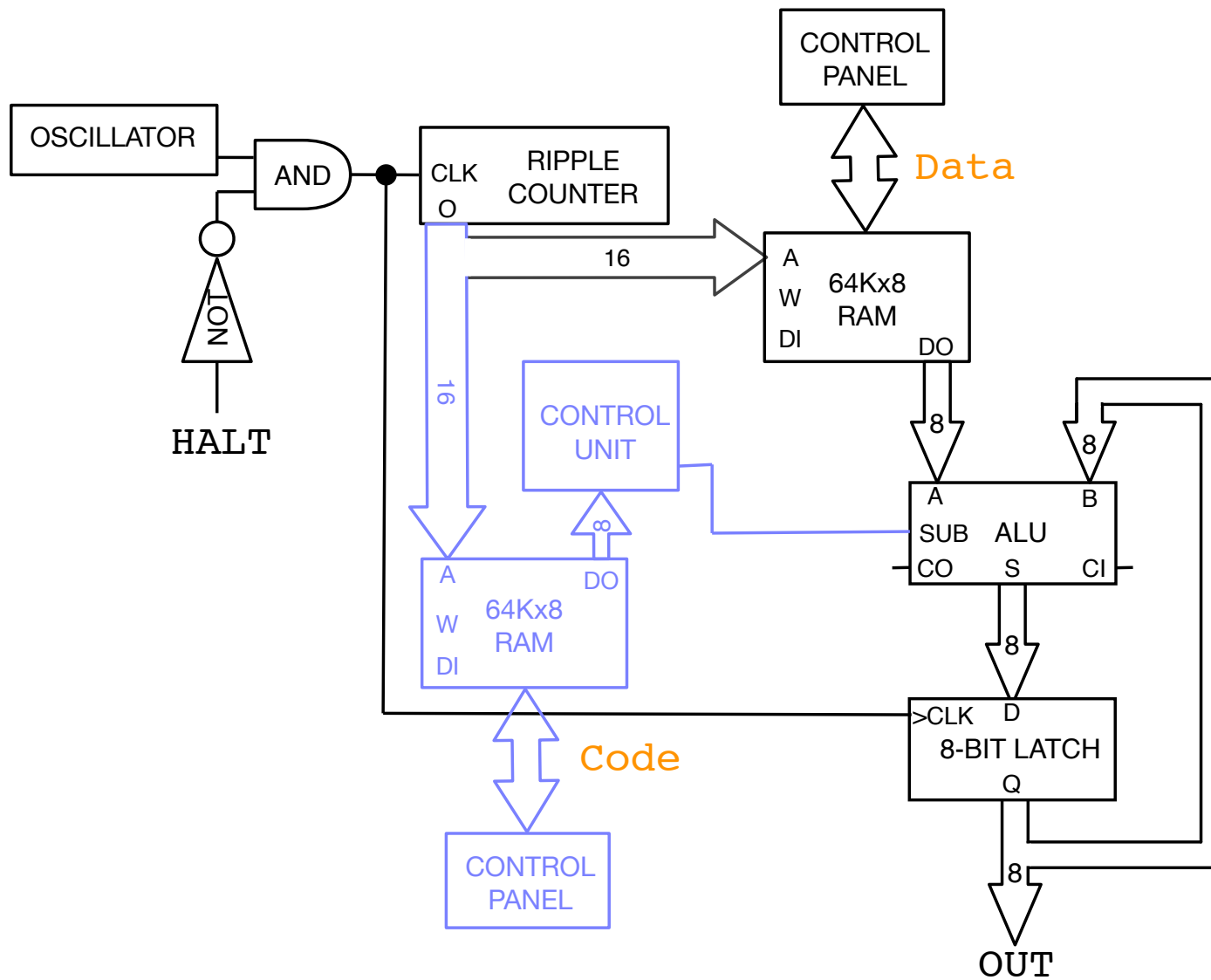
- Each operation is encoded by a byte value

Operation	Code (hex)
Add	20h
Subtract	21h

- Each operation is encoded by a byte value

Operation	Code (hex)
Add	20h
Subtract	21h

- Example

Address	Code	Data	Accumulator
0000h	20h Add	01h	
0001h	20h Add	02h	
0002h	21h Subtract	01h	
0003h	20h Add	08h	
0004h	21h Subtract	03h	

- Each operation is encoded by a byte value

Operation	Code (hex)
Add	20h
Subtract	21h

- Example

Address	Code	Data	Accumulator
			00h
0000h	20h Add	01h	
0001h	20h Add	02h	
0002h	21h Subtract	01h	
0003h	20h Add	08h	
0004h	21h Subtract	03h	

- Each operation is encoded by a byte value

Operation	Code (hex)
Add	20h
Subtract	21h

- Example

Address	Code	Data	Accumulator
			00h
0000h	20h Add	01h	01h
0001h	20h Add	02h	
0002h	21h Subtract	01h	
0003h	20h Add	08h	
0004h	21h Subtract	03h	

- Each operation is encoded by a byte value

Operation	Code (hex)
Add	20h
Subtract	21h

- Example

Address	Code	Data	Accumulator
			00h
0000h	20h Add	01h	01h
0001h	20h Add	02h	03h
0002h	21h Subtract	01h	
0003h	20h Add	08h	
0004h	21h Subtract	03h	

- Each operation is encoded by a byte value

Operation	Code (hex)
Add	20h
Subtract	21h

- Example

Address	Code	Data	Accumulator
			00h
0000h	20h Add	01h	01h
0001h	20h Add	02h	03h
0002h	21h Subtract	01h	02h
0003h	20h Add	08h	
0004h	21h Subtract	03h	

- Each operation is encoded by a byte value

Operation	Code (hex)
Add	20h
Subtract	21h

- Example

Address	Code	Data	Accumulator
			00h
0000h	20h Add	01h	01h
0001h	20h Add	02h	03h
0002h	21h Subtract	01h	02h
0003h	20h Add	08h	0ah
0004h	21h Subtract	03h	

Operation Codes for Instructions

- Each operation is encoded by a byte value

Operation	Code (hex)
Add	20h
Subtract	21h

- Example

Address	Code	Data	Accumulator
			00h
0000h	20h Add	01h	01h
0001h	20h Add	02h	03h
0002h	21h Subtract	01h	02h
0003h	20h Add	08h	0ah
0004h	21h Subtract	03h	07h

Operation Codes for Instructions

- Each operation is encoded by a byte value

Operation	Code (hex)
Add	20h
Subtract	21h

- Example

Address	Code	Data	Accumulator
			00h
0000h	20h Add	01h	01h
0001h	20h Add	02h	03h
0002h	21h Subtract	01h	02h
0003h	20h Add	08h	0ah
0004h	21h Subtract	03h	07h

More Instructions

- Load: load number from memory into accumulator

- Store: store accumulator value in memory

- Halt: block clock

- Each operation is encoded by a byte value

Operation	Code (hex)
Load	10h
Store	11h
Add	20h
Subtract	21h
Halt	FFh

More Wiring
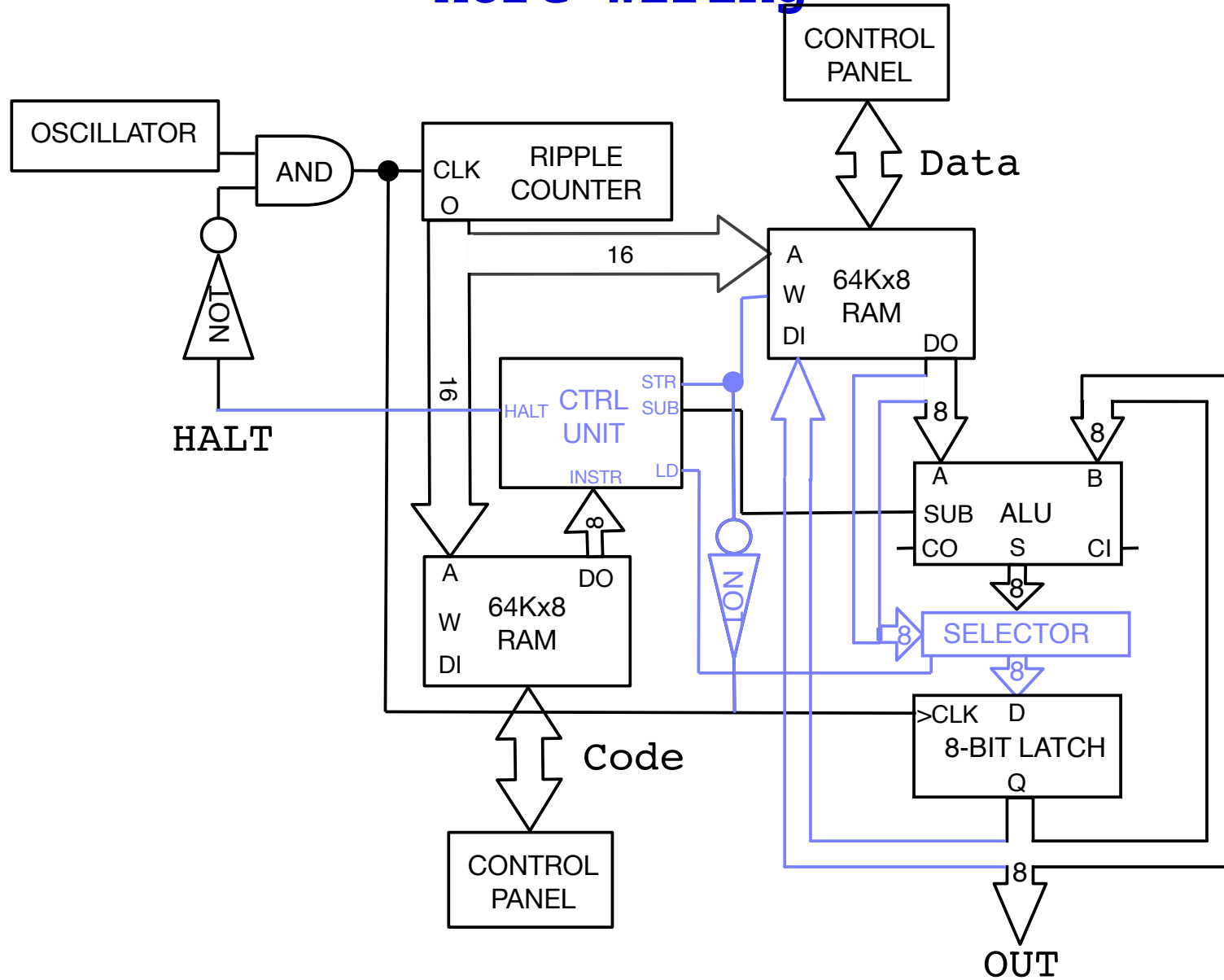
Load
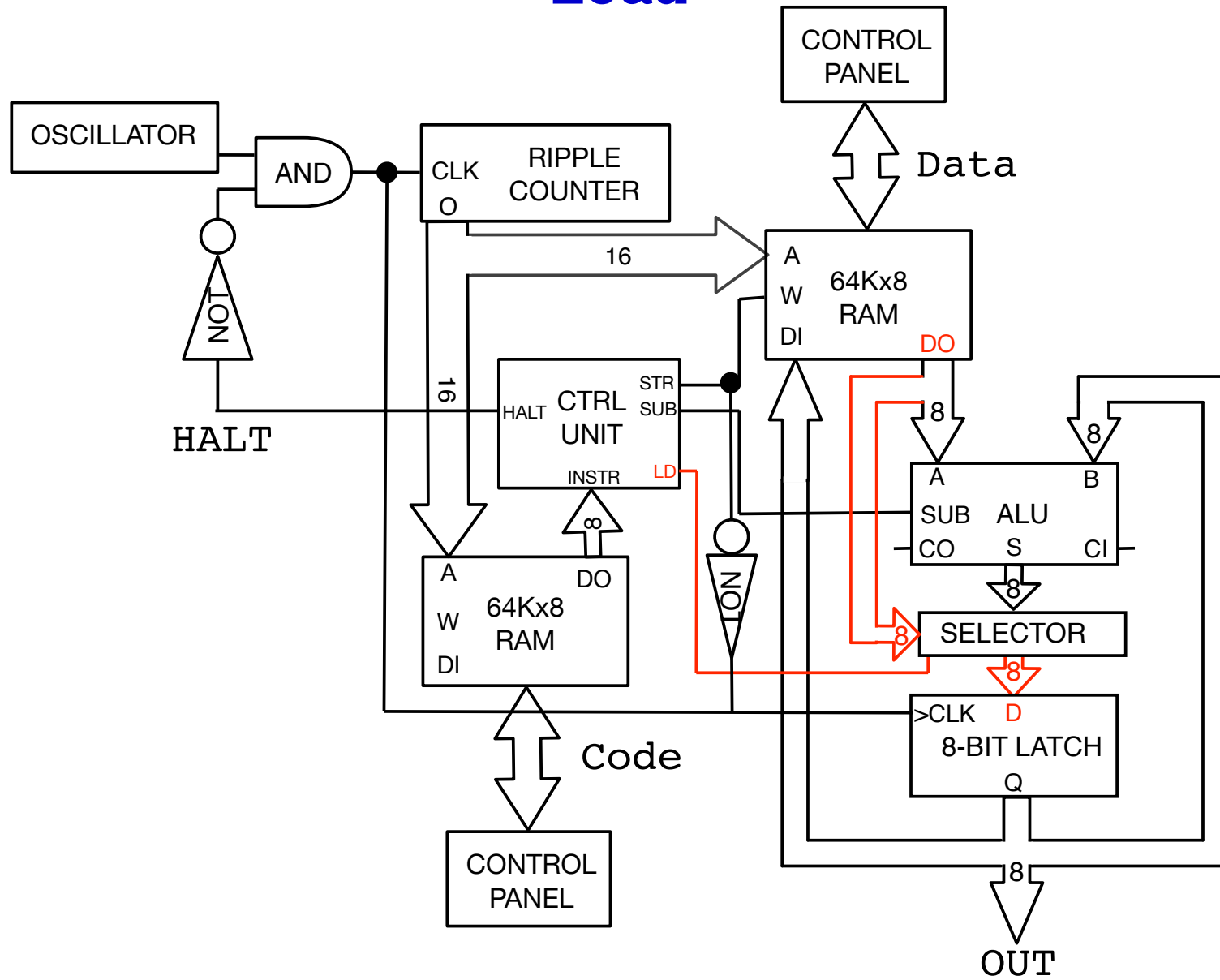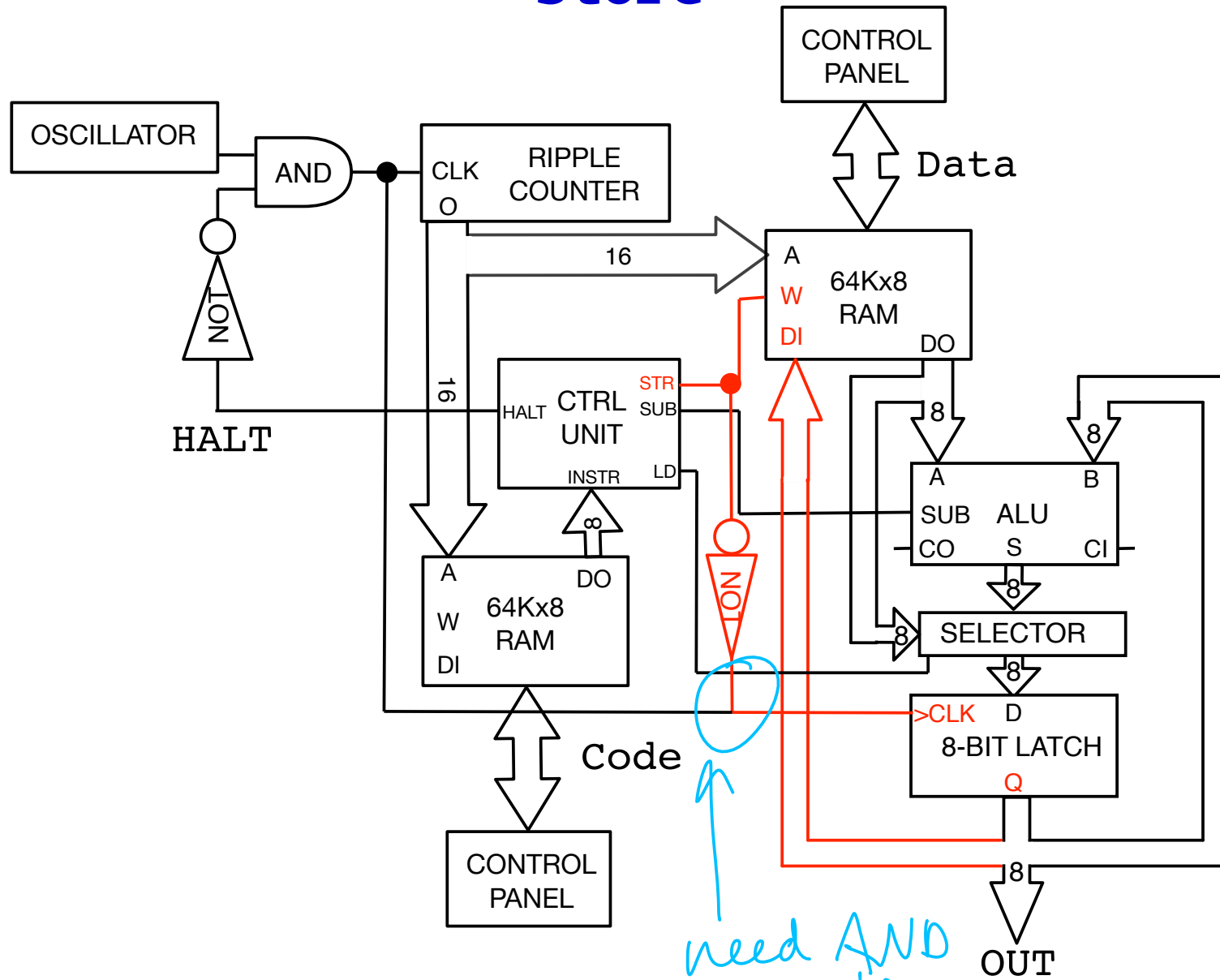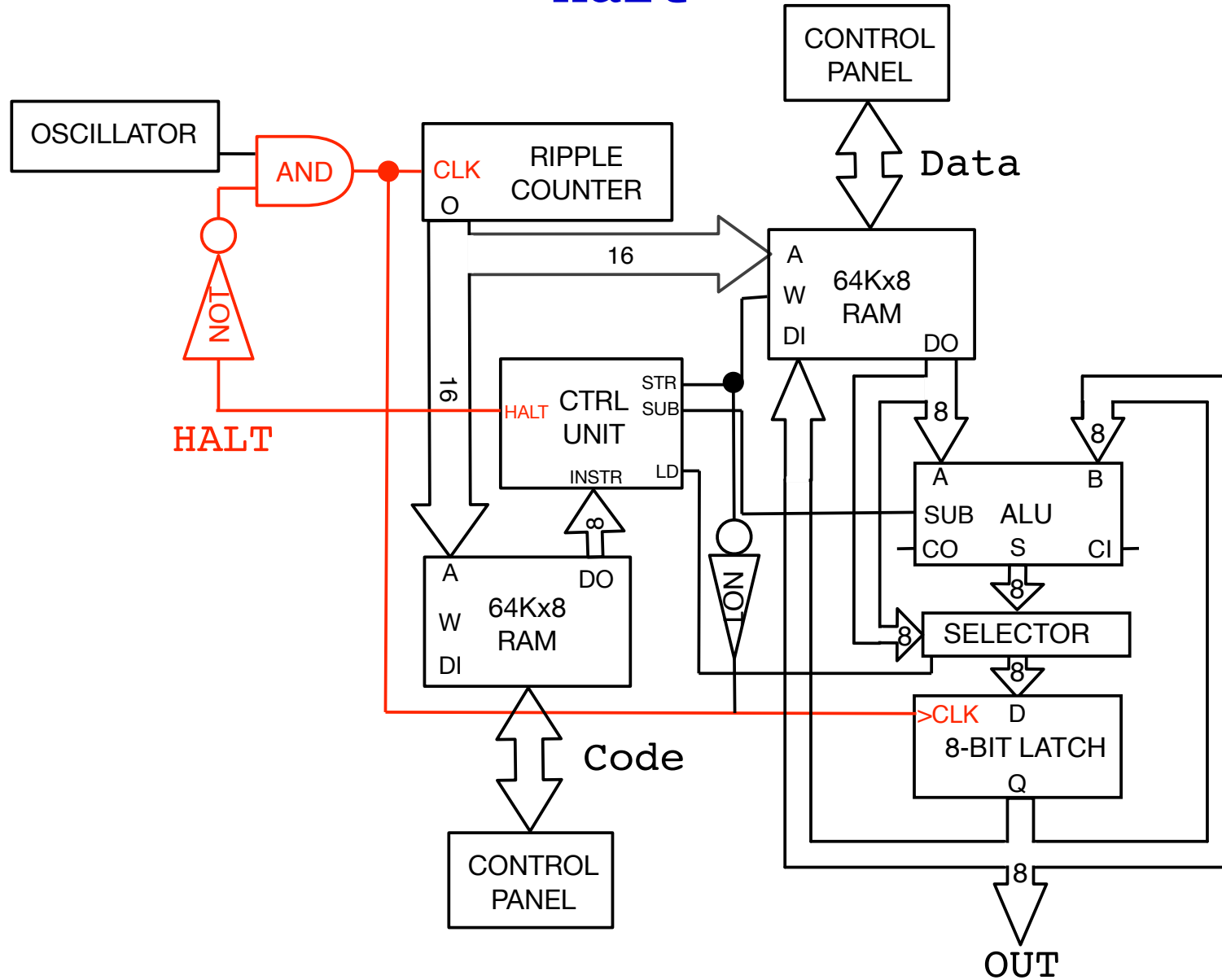
Operations and Wiring

- Each operation changes certain flags

Operation	Code (hex)	LD	STR	SUB	HALT
Load	10h	1	0	0	0
Store	11h	0	1	0	0
Add	20h	0	0	0	0
Subtract	21h	0	0	1	0
Halt	FFh	0	0	0	1

Program Example

Address	Code		Data	Accumulator
0000h	10h	Load	56h	
0001h	20h	Add	2Ah	
0002h	21h	Subtract	38h	
0003h	11h	Store		
0004h	FFh	Halt		

Program Example

```
Address       Code       Data    Accumulator
                                     00h

0000h     10h Load       56h
0001h     20h Add        2Ah
0002h     21h Subtract   38h
0003h     11h Store
0004h     FFh Halt
```

Program Example

Address	Code		Data	Accumulator
				00h
0000h	10h	Load	56h	56h
0001h	20h	Add	2Ah	
0002h	21h	Subtract	38h	
0003h	11h	Store		
0004h	FFh	Halt		

Program Example

```
Address        Code        Data    Accumulator
                                      00h
0000h     10h Load        56h       56h
0001h     20h Add         2Ah       80h
0002h     21h Subtract    38h
0003h     11h Store
0004h     FFh Halt
```

Program Example

Address	Code		Data	Accumulator
				00h
0000h	10h	Load	56h	56h
0001h	20h	Add	2Ah	80h
0002h	21h	Subtract	38h	48h
0003h	11h	Store		
0004h	FFh	Halt		

Program Example

```
Address        Code        Data    Accumulator
                                    00h
0000h    10h Load          56h     56h
0001h    20h Add           2Ah     80h
0002h    21h Subtract      38h     48h
0003h    11h Store                 48h
0004h    FFh Halt
```

Program Example

Address	Code		Data	Accumulator
				00h
0000h	10h	Load	56h	56h
0001h	20h	Add	2Ah	80h
0002h	21h	Subtract	38h	48h
0003h	11h	Store		48h
0004h	FFh	Halt		48h

Address	Code		Data	Accumulator
				00h
0000h	10h	Load	56h	56h
0001h	20h	Add	2Ah	80h
0002h	21h	Subtract	38h	48h
0003h	11h	Store		48h
0004h	FFh	Halt		48h

adding 16 bit numbers

Adding 16 Bit Numbers

- 1 byte integers will not suffice in practice

  - unsigned: 0 to 255
  - signed: -128 to 127

- Let's use 2 bytes (16 bit)

- How can we do addition with our 8-bit adder?

Adding 16 Bit Numbers

- 1 byte integers will not suffice in practice

  – unsigned: 0 to 255
  – signed: -128 to 127

- Let's use 2 bytes (16 bit)

- How can we do addition with our 8-bit adder?

  Add the bytes separately

Example

- Task: 76ABh + 232Ch

- Lower order byte

$$
\begin{array}{r}
\text{ABh} \\
+\text{2Ch} \\
\hline
\text{D7h}
\end{array}
$$

- Higher order byte

$$
\begin{array}{r}
\text{76h} \\
+\text{23h} \\
\hline
\text{99h}
\end{array}
$$

- Putting it together: 99D7h

Another Example

- Task: 76ABh + 236Ch

- Lower order byte

```
  ABh
+6Ch
----
117h
```

- Higher order byte
  (add the carry)

1h ⟵ *carry*

```
1h
+76h
+23h
----
 9Ah
```

- Putting it together: 9AD7h

- Add with Carry

  − when addition results in a carry, store this in a flag

  − new add instruction that includes carry if flag set

More Instructions

- Add with Carry

  – when addition results in a carry, store this in a flag

  – new add instruction that includes carry if flag set

- Subtract with Borrow

  – when subtraction results in a carry, store this in a flag

  – new subtract instruction that includes carry if flag set
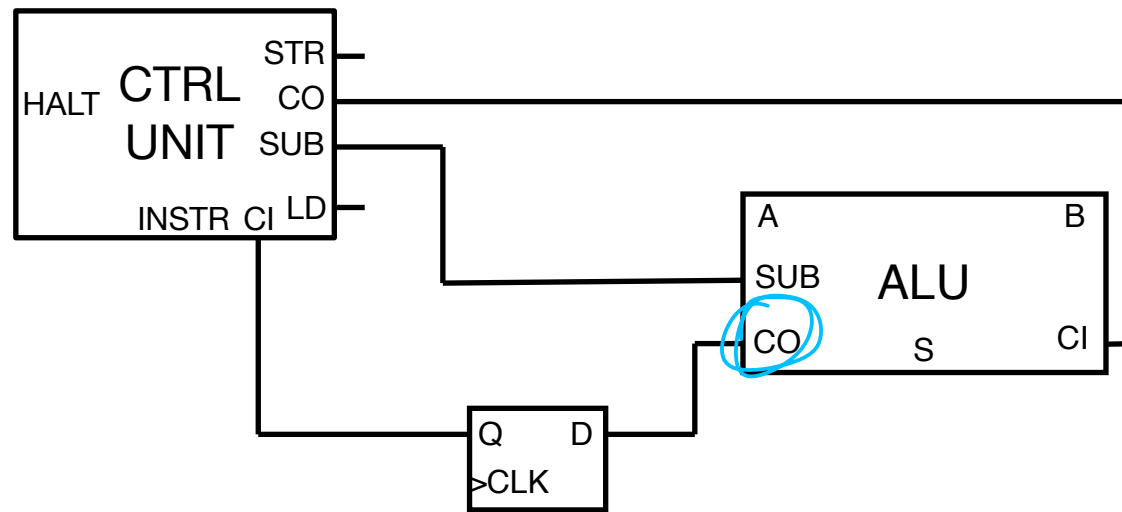
Instructions

- Each operation is encoded by a byte value

Operation	Code (hex)
Load	10h
Store	11h
Add	20h
Subtract	21h
Add with carry	22h
Subtract with borrow	23h
Halt	FFh

Address	Code	Data	Carry	Accumulator
0000h	10h Load	ABh		
0001h	20h Add	6Ch		
0002h	11h Store			
0003h	10h Load	76h		
0004h	22h Add with carry	23h		
0005h	11h Store			
0006h	FFh Halt			

Code

Address	Code	Data	Carry	Accumulator
			0	00h
0000h	10h Load	ABh		
0001h	20h Add	6Ch		
0002h	11h Store			
0003h	10h Load	76h		
0004h	22h Add with carry	23h		
0005h	11h Store			
0006h	FFh Halt			

Address	Code	Data	Carry	Accumulator
			0	00h
0000h	10h Load	ABh	0	ABh
0001h	20h Add	6Ch		
0002h	11h Store			
0003h	10h Load	76h		
0004h	22h Add with carry	23h		
0005h	11h Store			
0006h	FFh Halt			

Code

Address	Code	Data	Carry	Accumulator
			0	00h
0000h	10h Load	ABh	0	ABh
0001h	20h Add	6Ch	1	17h
0002h	11h Store			
0003h	10h Load	76h		
0004h	22h Add with carry	23h		
0005h	11h Store			
0006h	FFh Halt			

Code

Address	Code	Data	Carry	Accumulator
			0	00h
0000h	10h Load	ABh	0	ABh
0001h	20h Add	6Ch	1	17h
0002h	11h Store		1	17h
0003h	10h Load	76h		
0004h	22h Add with carry	23h		
0005h	11h Store			
0006h	FFh Halt			

Code

Address	Code	Data	Carry	Accumulator
			0	00h
0000h	10h Load	ABh	0	ABh
0001h	20h Add	6Ch	1	17h
0002h	11h Store		1	17h
0003h	10h Load	76h	1	76h
0004h	22h Add with carry	23h		
0005h	11h Store			
0006h	FFh Halt			

Code

Address	Code	Data	Carry	Accumulator
			0	00h
0000h	10h Load	ABh	0	ABh
0001h	20h Add	6Ch	1	17h
0002h	11h Store		1	17h
0003h	10h Load	76h	1	76h
0004h	22h Add with carry	23h	0	9Ah
0005h	11h Store			
0006h	FFh Halt			

"condition codes"
carry, borrow, etc.

Code

Address	Code	Data	Carry	Accumulator
			0	00h
0000h	10h Load	ABh	0	ABh
0001h	20h Add	6Ch	1	17h
0002h	11h Store		1	17h
0003h	10h Load	76h	1	76h
0004h	22h Add with carry	23h	0	9Ah
0005h	11h Store		0	9Ah
0006h	FFh Halt			

Code

Address	Code		Data	Carry	Accumulator
				0	00h
0000h	10h	Load	ABh	0	ABh
0001h	20h	Add	6Ch	1	17h
0002h	11h	Store		1	17h
0003h	10h	Load	76h	1	76h
0004h	22h	Add with carry	23h	0	9Ah
0005h	11h	Store		0	9Ah
0006h	FFh	Halt		0	9Ah

35

Code

Address	Code		Data	Carry	Accumulator
				0	00h
0000h	10h	Load	ABh	0	ABh
0001h	20h	Add	6Ch	1	17h
0002h	11h	Store		1	17h
0003h	10h	Load	76h	1	76h
0004h	22h	Add with carry	23h	0	9Ah
0005h	11h	Store		0	9Ah
0006h	FFh	Halt		0	9Ah

addressing memory

Motivation

- Currently using two memories

  – code memory for instructions

  – data memory

- Very limiting

- Instead:

  – store code and data in same memory

  – add explicit addresses to instructions

Adapted 16-Bit Adder

- Memory

Address	Data
4000h	ABh
4001h	76h
4002h	6Ch
4003h	23h

loc. of data in memor (handwritten annotation)

data address (handwritten annotation)

- Code

Address	Bytes	Code
0000h	10h 00h 40h	Load 4000h
0003h	20h 02h 40h	Add 4002h
0006h	11h 04h 40h	Store 4004h
0009h	10h 01h 40h	Load 4001h
000Ch	22h 03h 40h	Add with carry 4003h
000Fh	11h 05h 40h	Store 4005h
0012h	FFh	Halt

- Memory

Address	Data
4000h	ABh
4001h	76h
4002h	6Ch
4003h	23h

- Code

Address	Bytes	Code
0000h	10h 00h 40h	Load 4000h
0003h	20h 02h 40h	Add 4002h
0006h	11h 04h 40h	Store 4004h
0009h	10h 01h 40h	Load 4001h
000Ch	22h 03h 40h	Add with carry 4003h
000Fh	11h 05h 40h	Store 4005h
0012h	FFh	Halt

Note:
Instructions
take up
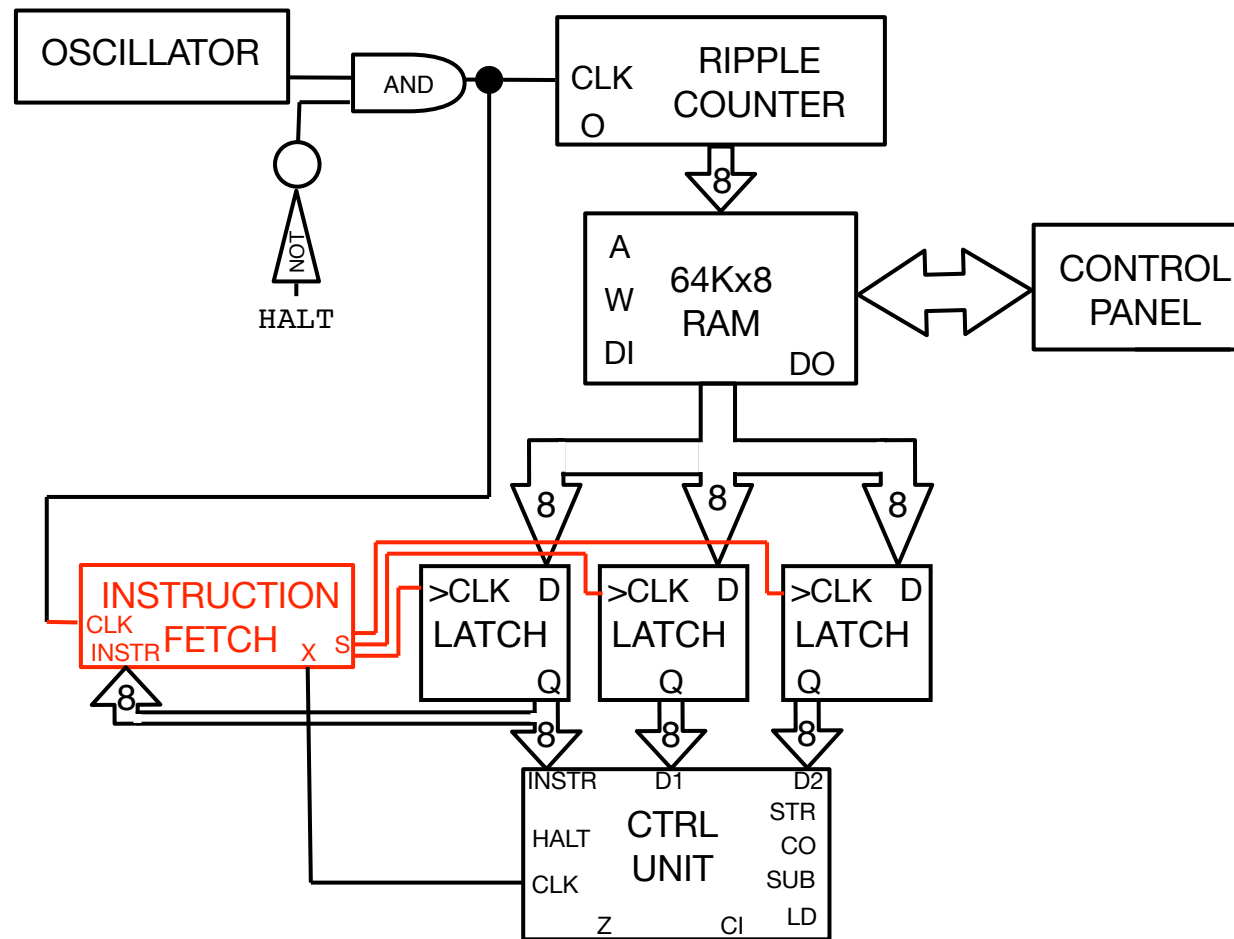1 or 3 bytes
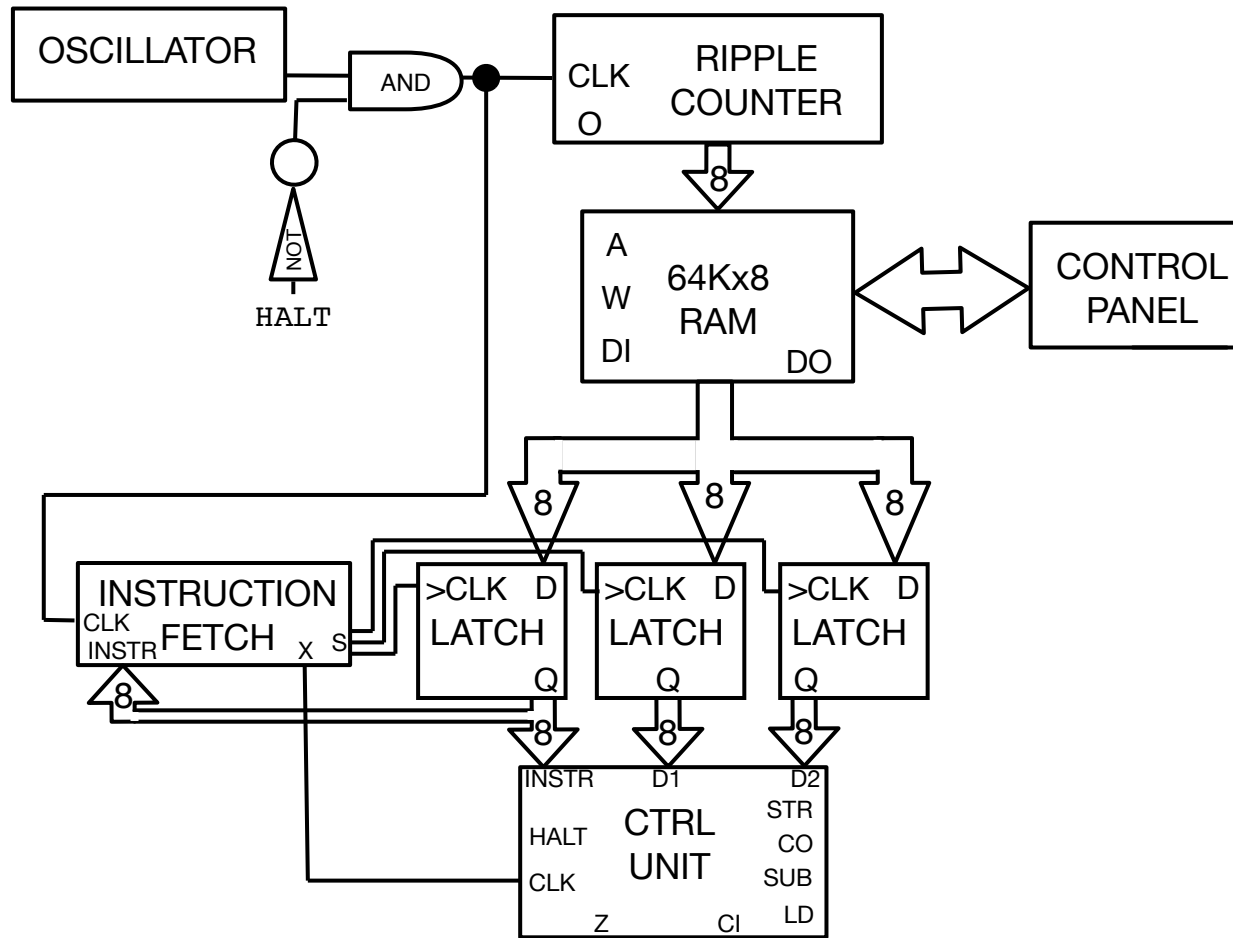
Instruction Fetch

3 registers: code and 2 byte data

Instruction Fetch
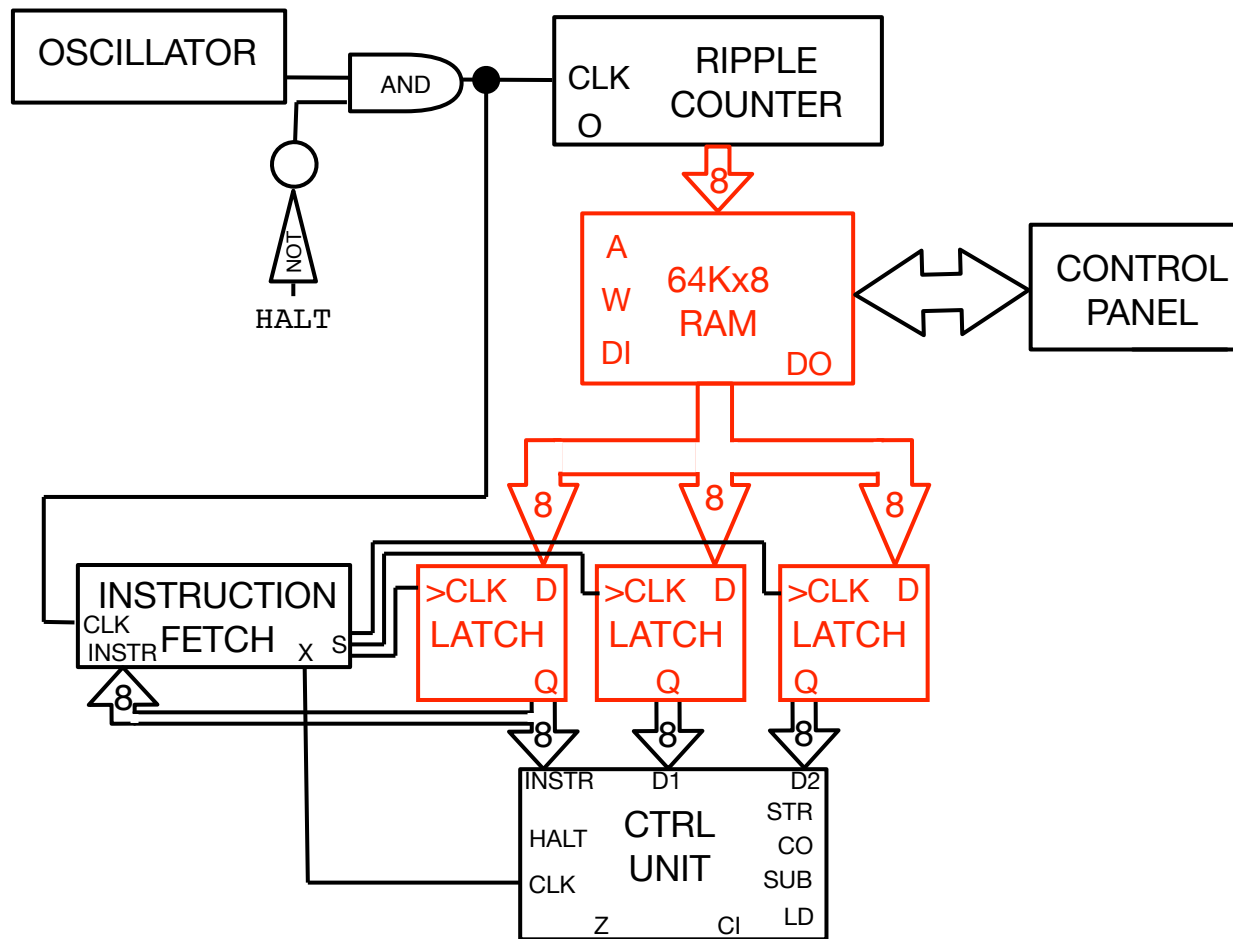
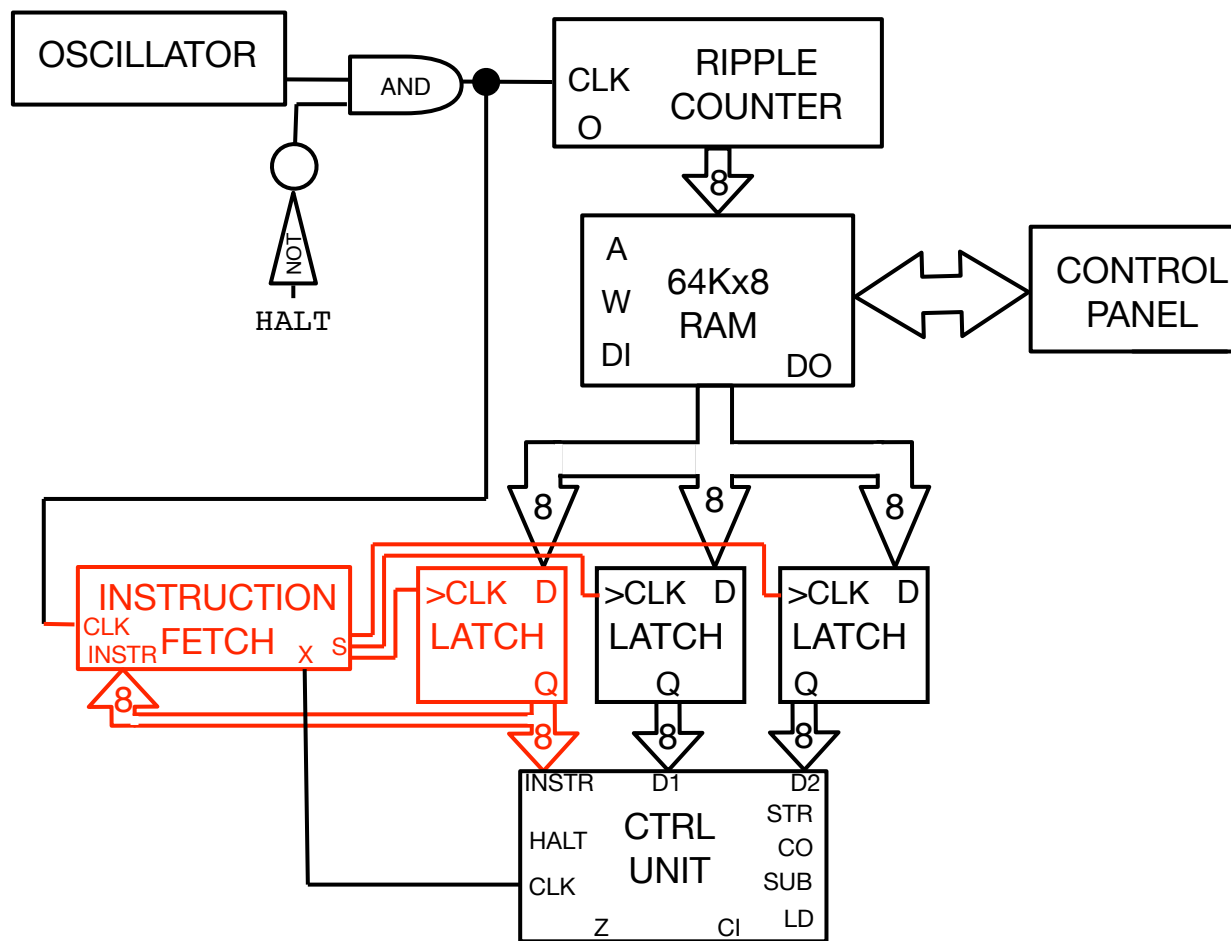

Transfer bytes from memory to instruction code and data register
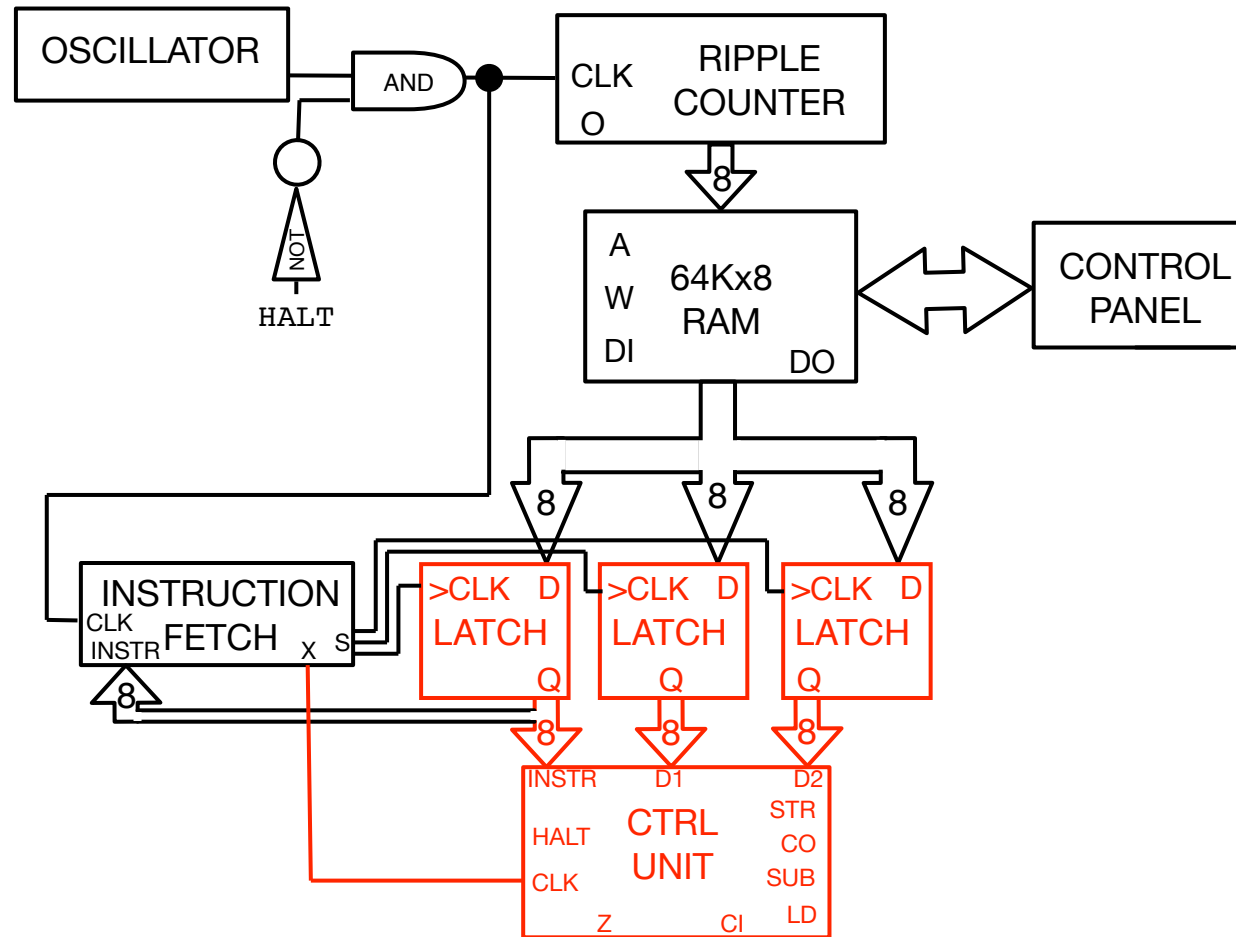
Instruction Fetch

Instruction fetch logic determines which register is written to

This is informed by instruction code

Instruction Fetch

Once all registers are filled, execute instruction

Data Paths

- Data needs to be transferred in various ways

- Adress passed on to memory (overriding program counter)

- Add/subtract: read byte from memory, pass to ALU

- Load: read byte from memory, store in accumulator

- Store: read byte from accumulator, store in memory

- No detailed wiring worked out here...

multiplication

- Multiplication by repeated addition

- Pseudo-code

```
load number1 into accumulator
loop
    subtract 1 from number2
    last if number2 = 0
    add number1 to accumulator
store accumulator in result
```

- Jump

   – set the ripple counter to specified value

- Jump

  – set the ripple counter to specified value

- Zero flag

  – detect that subtraction resulted in number 0
  – implemented as flag of the ALU

Needed

- Jump

  – set the ripple counter to specified value

- Zero flag

  – detect that subtraction resulted in number 0
  – implemented as flag of the ALU

- Jump if zero

  – check if zero flag is set
  – only then update ripple counter
  – otherwise, do nothing

Zero Flag
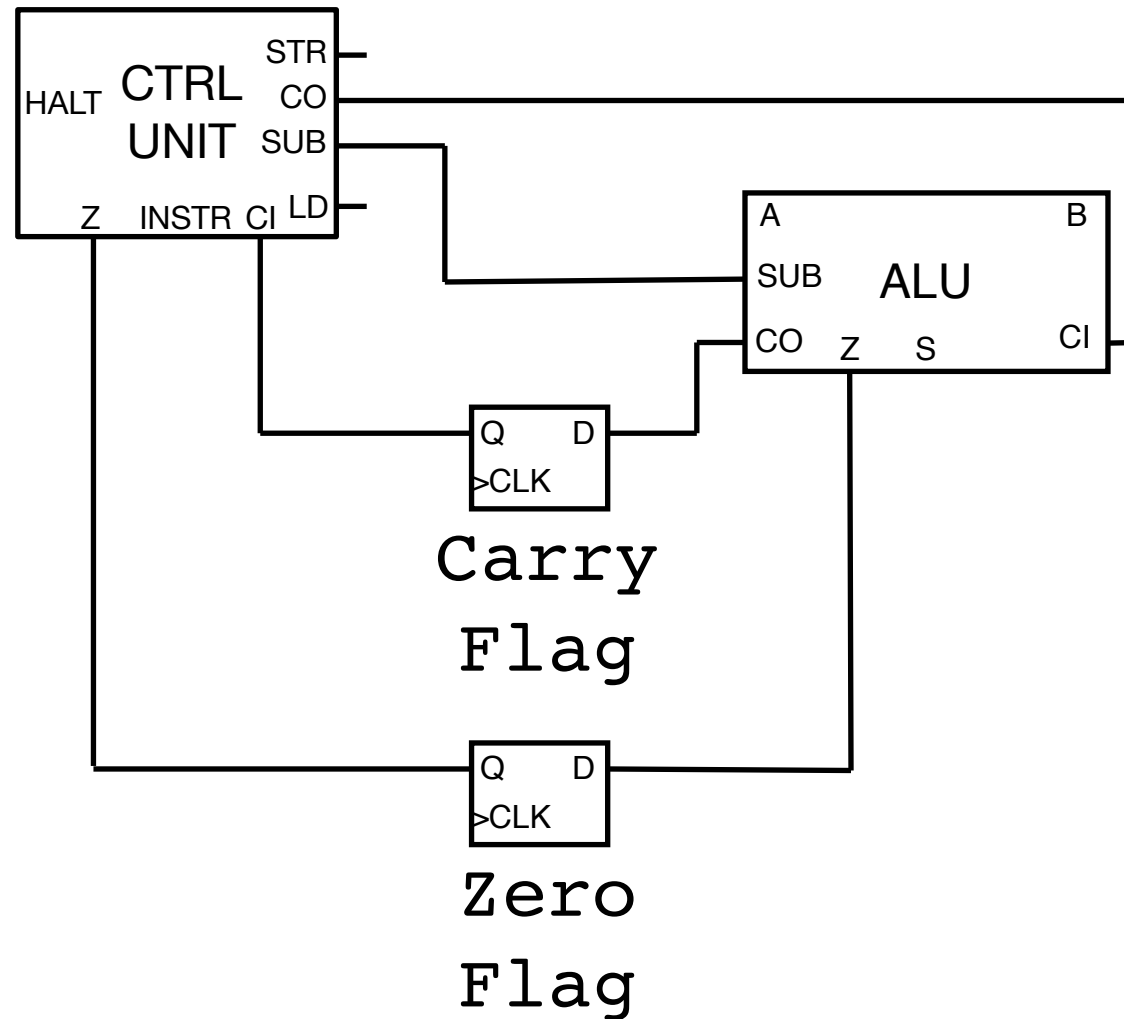
- Flag when the ALU operation results in 0

Operation	Code (hex)
Load	10h
Store	11h
Add	20h
Subtract	21h
Add with carry	22h
Subtract with borrow	23h
Jump	30h
Jump if zero	31h
Jump if carry	32h
Jump if not zero	33h
Jump if not carry	34h
Halt	FFh

Code (8 Bit Version)

- Memory

Address	Data
4000h	0Bh
4001h	0Fh
4002h	01h

- Code

Address	Bytes	Code
0000h	10h 00h 40h	Load 4000h ; load number1
0003h	11h 03h 40h	Store 4003h ; save in result
0006h	10h 01h 40h	Load 4001h ; load number2
0009h	21h 02h 40h	Subtract 4002h ; subtract 1
000Bh	31h 18h 00h	jump if zero to 0018h ; jump to end if done
000Fh	10h 03h 40h	load 4003h ; load result
0012h	20h 00h 40h	add 4000h ; add number1
0015h	30h 03h 00h	jump to 0003h ; loop
0018h	FFh	halt ; quit