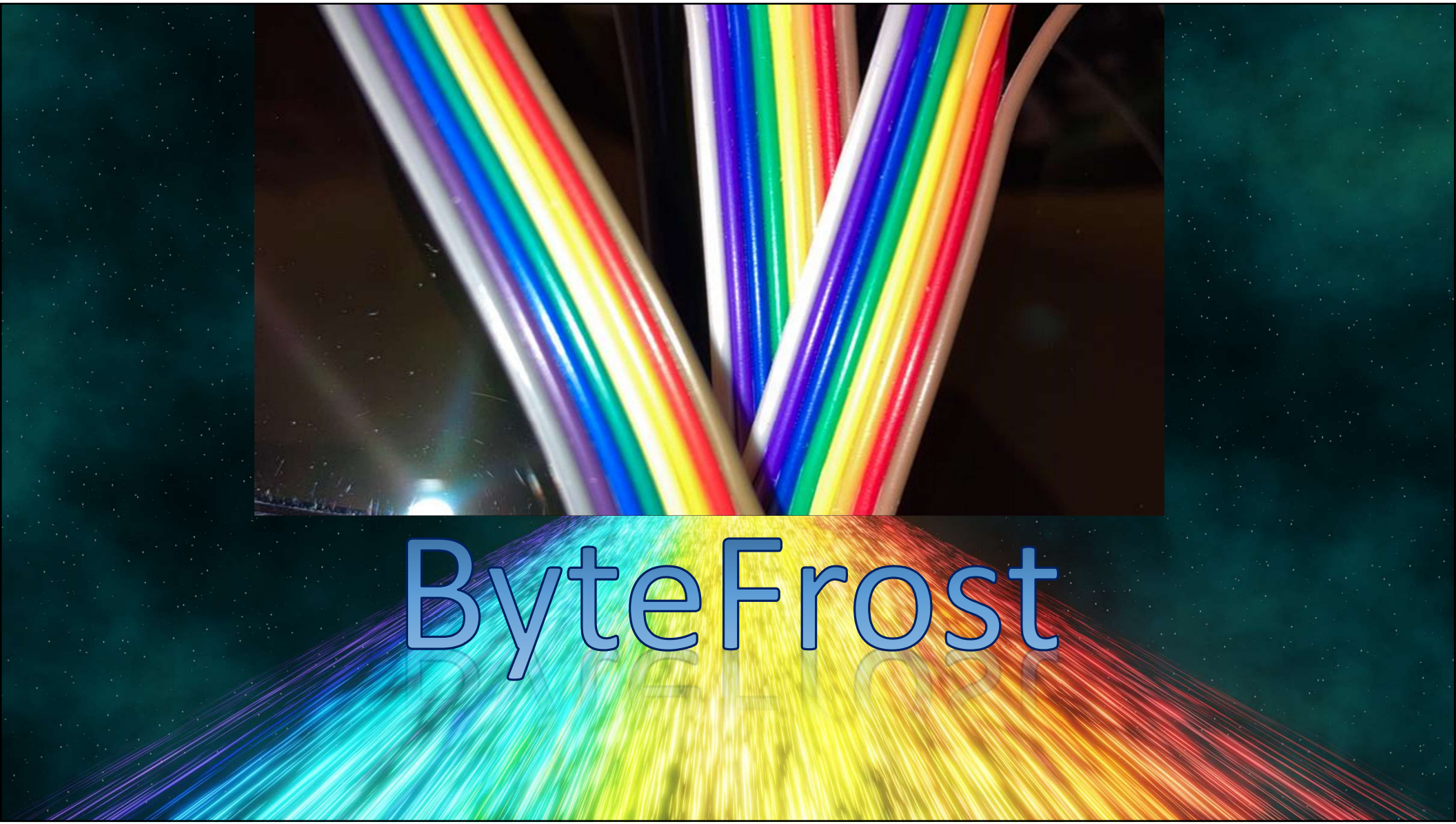
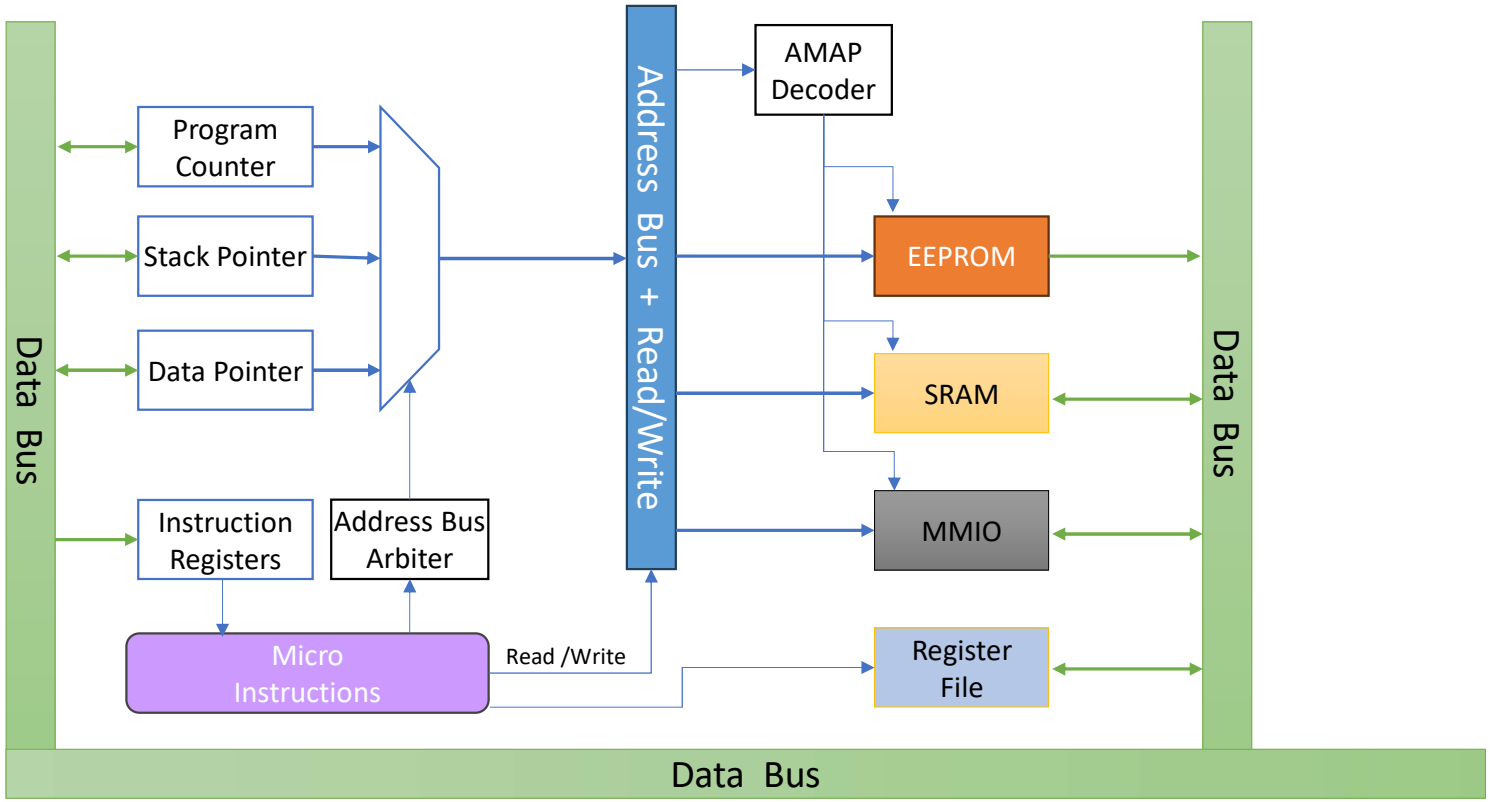
And
the
Name
…

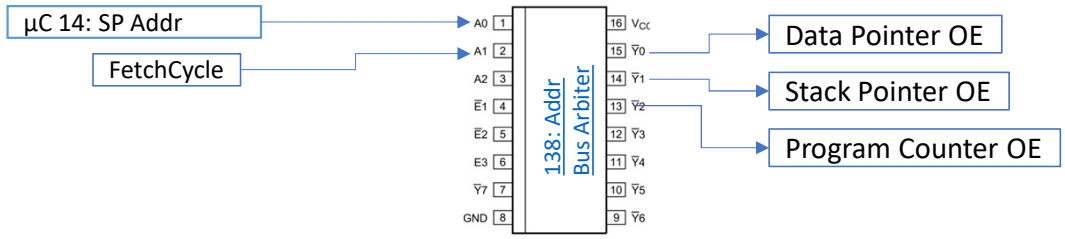

ByteFrost

ByteFrost V2.0 – 16bit Address

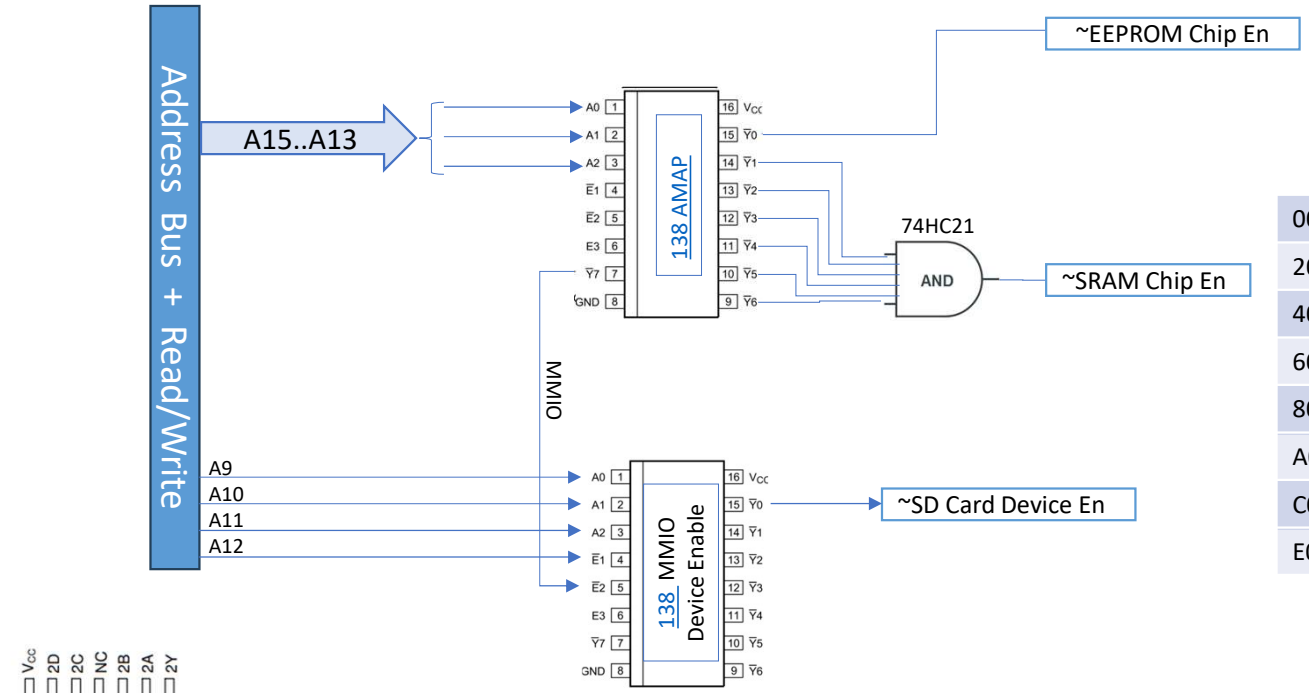
Unified Bus Operation (v2.0)



Address Bus Arbiter (v2.0)



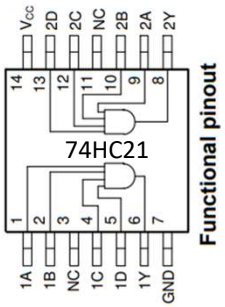
Address Map Decoder (v2.0)



Address Map

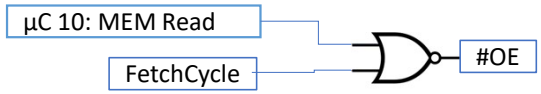
0000 – 1FFF	ROM
2000 – 3FFF	RAM
4000 – 5FFF	RAM
6000 – 7FFF	RAM
8000 – 9FFF	RAM
A000 – BFFF	RAM
C000 – DFFF	RAM
E000 – FFFF	MMIO

E000 – E1FF	SD Card
E200 – E3FF	
E400 – E5FF	
E600 – E7FF	
E800 – E9FF	
EA00 – EBFF	
EC00 – EDFF	
EE00 – EFFF	
F000 – FFFF	No HW

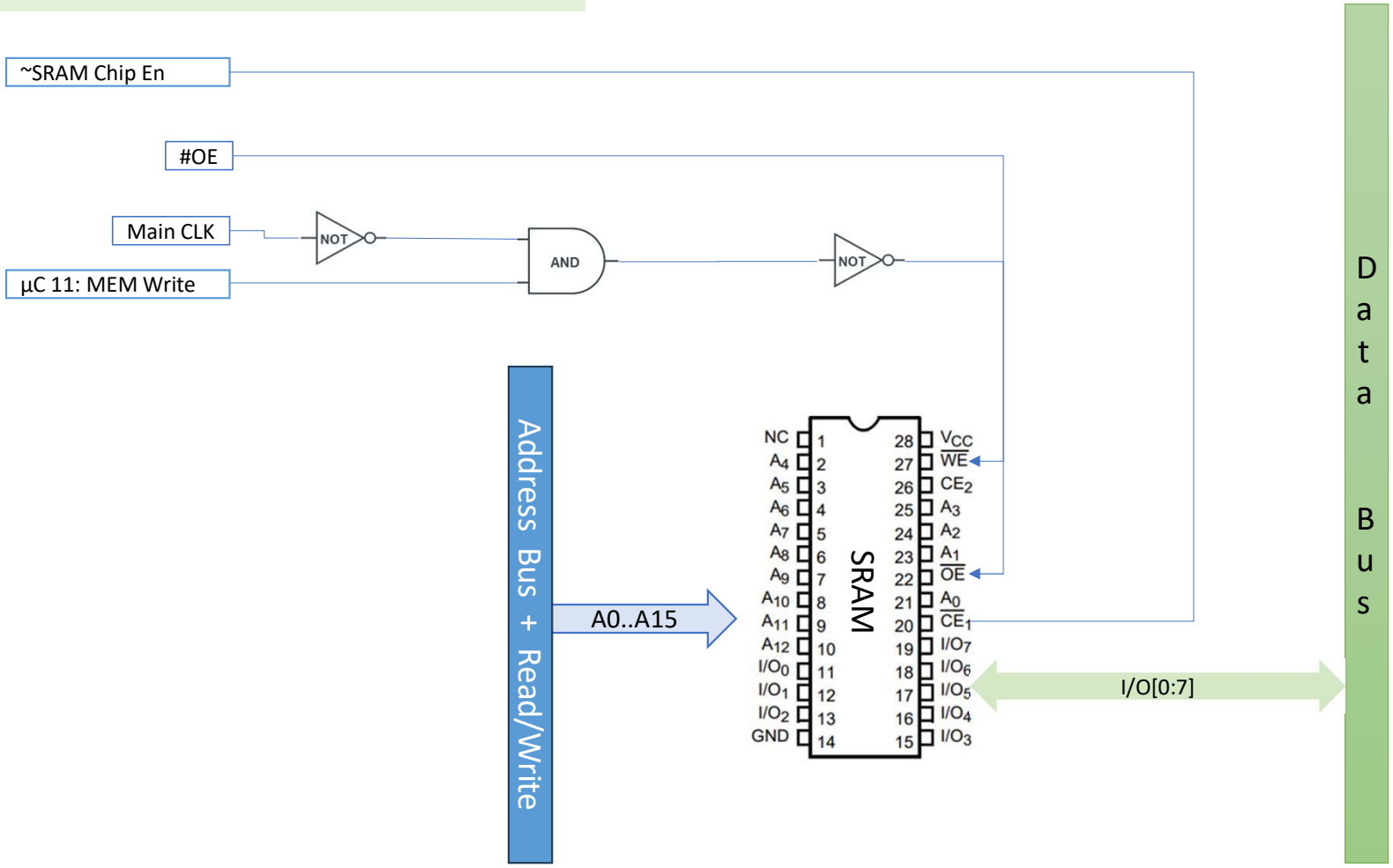


Memory Access (v2.0)

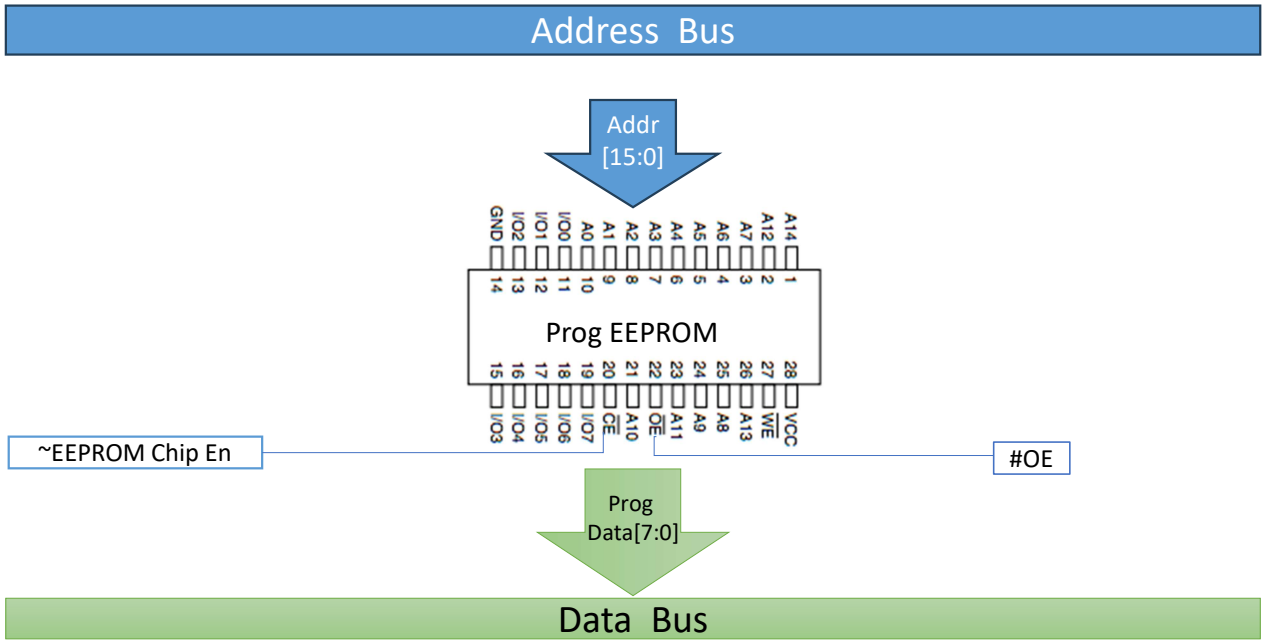
Signal	SRAM	EEPROM
#WE	(µC 11: MEM Write)	'1' (no writes)
#OE	(FetchCycle) NOR (µC 10: MEM Read)	
#CE	(~SRAM Chip En)	(~EEPROM Chip En)



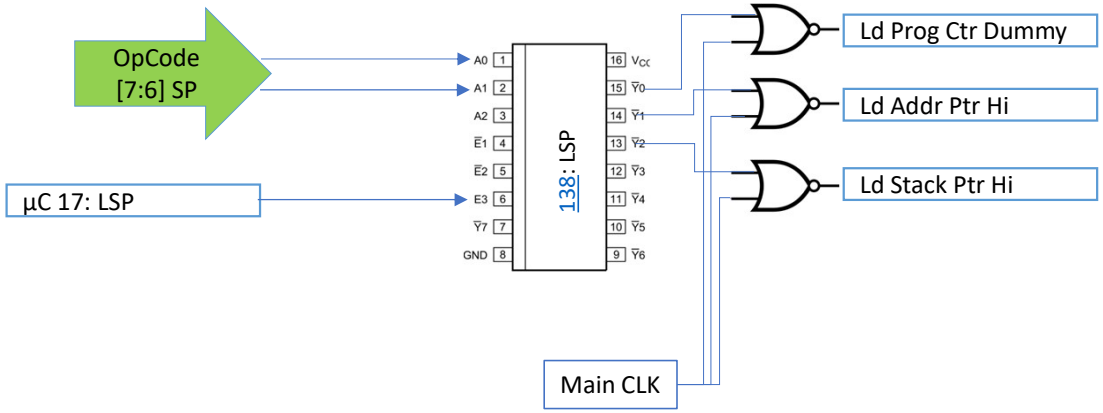
SRAM (v2.0)

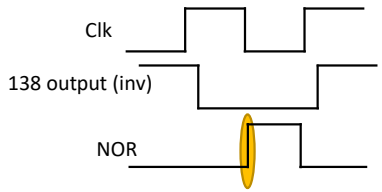

~SRAM Chip En
#OE
Main CLK
µC 11: MEM Write
NOT
AND
NOT
Address Bus + Read/Write
A0..A15
SRAM
NC 1
A4 2
A5 3
A6 4
A7 5
A8 6
A9 7
A10 8
A11 9
A12 10
I/O0 11
I/O1 12
I/O2 13
GND 14
28 VCC
27 WE
26 CE2
25 A3
24 A2
23 A1
22 OE
21 A0
20 CE1
19 I/O7
18 I/O6
17 I/O5
16 I/O4
15 I/O3
I/O[0:7]
Data Bus

EEPROM (v2.0)


Address Bus
Addr
[15:0]
GND I/O2 I/O1 I/O0 A0 A1 A2 A3 A4 A5 A6 A7 A12 A14
14 13 12 11 10 9 8 7 6 5 4 3 2 1
Prog EEPROM
15 16 17 18 19 20 21 22 23 24 25 26 27 28
I/O3 I/O4 I/O5 I/O6 I/O7 CE A10 OE A11 A9 A8 A13 WE VCC
~EEPROM Chip En
#OE
Prog
Data[7:0]
Data Bus

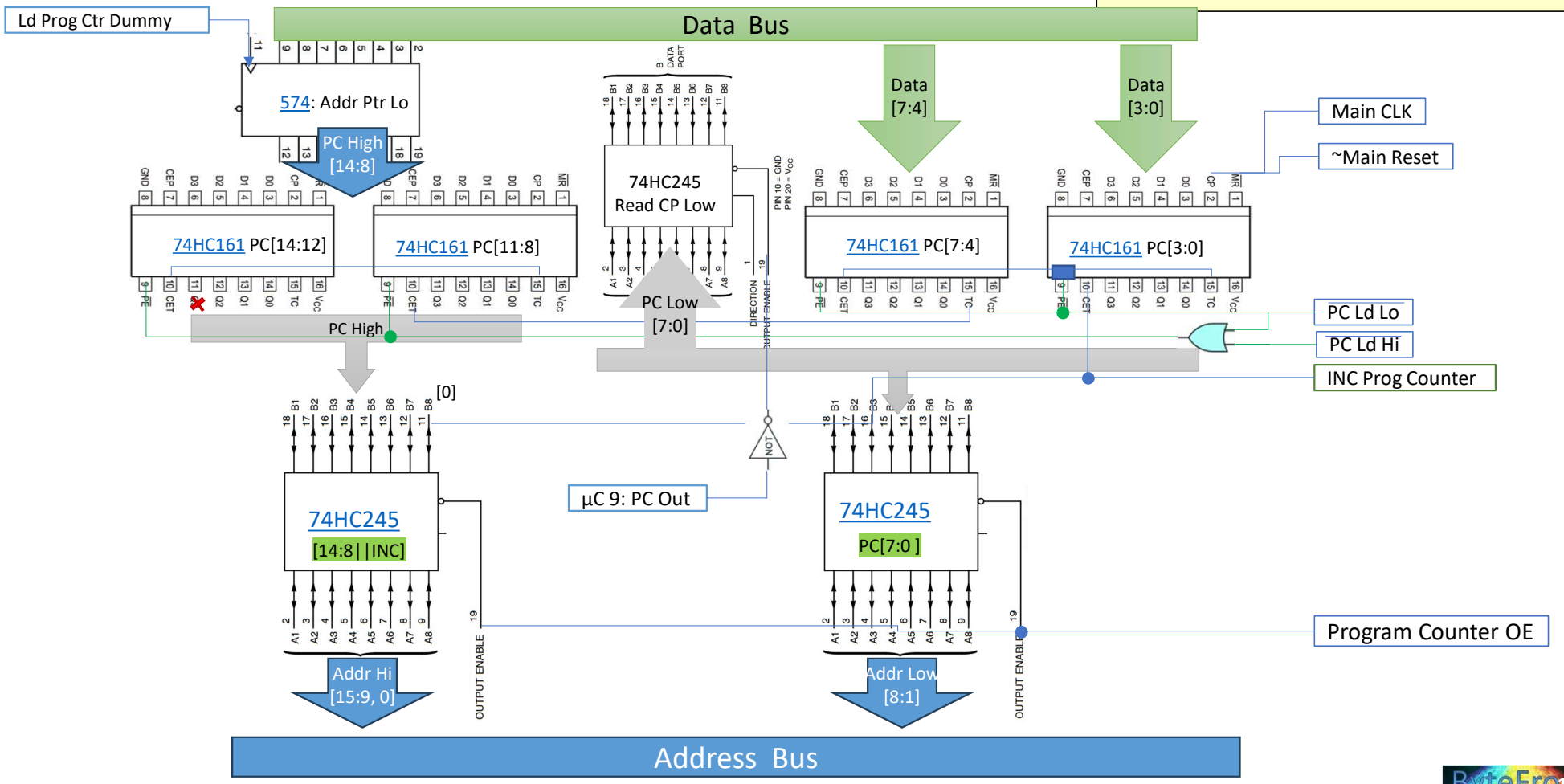
Load Special Pointer (LSP) (v2.0)



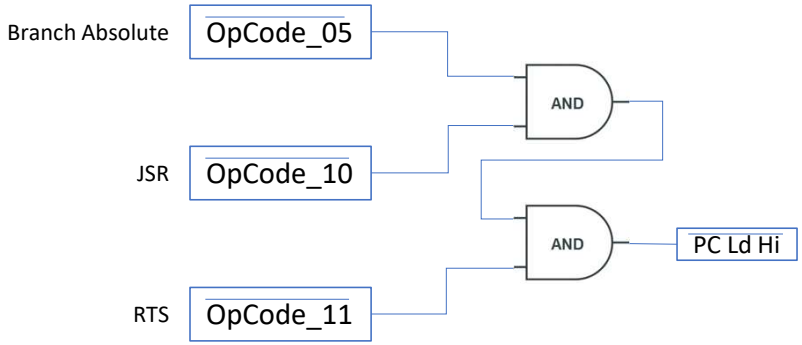


Program Counter (v2.0)

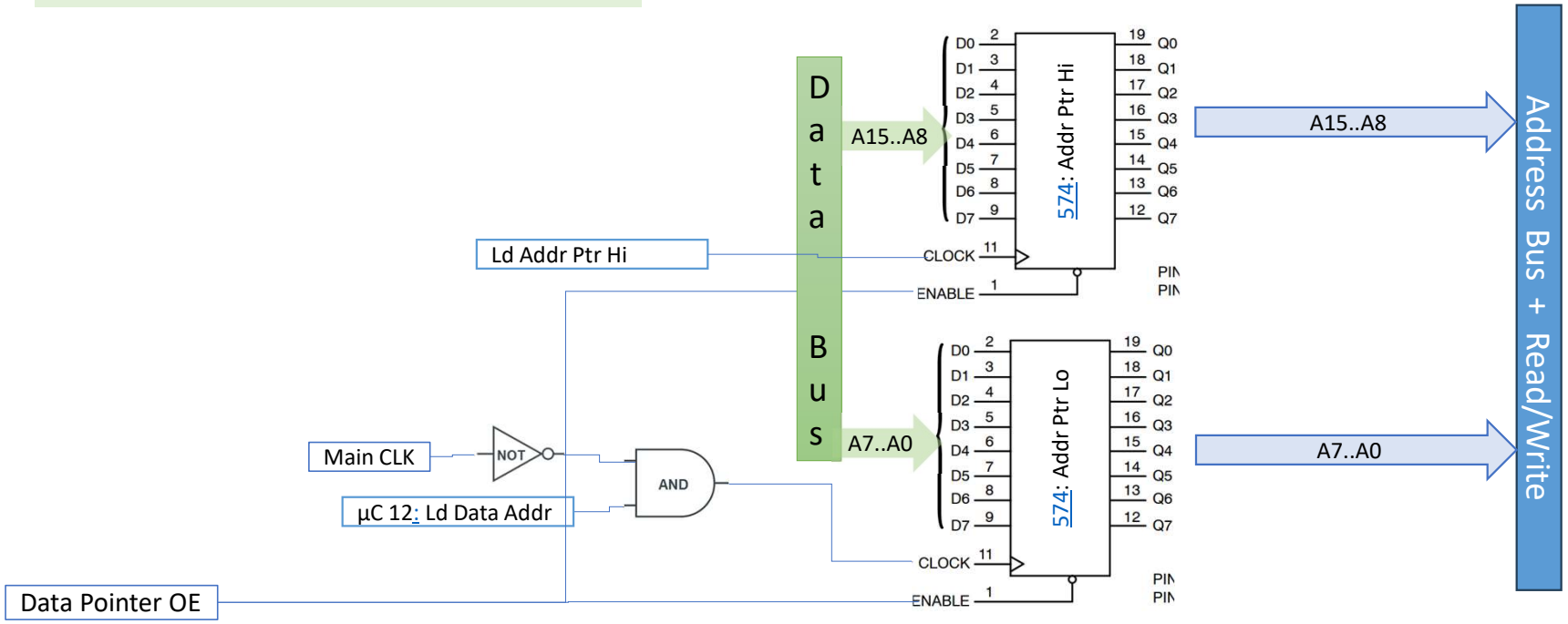
- Prog Counter 14:0 are mapped to Address Bus 15:1.
- "INC Prog Counter" is mapped to Address Bus bit 0
- Relative Branch only update PC 7:0, or Address bus 8:1
- Absolute Branch, JSR, RTS use the PC Dummy Reg
- PC bit 15 is not used

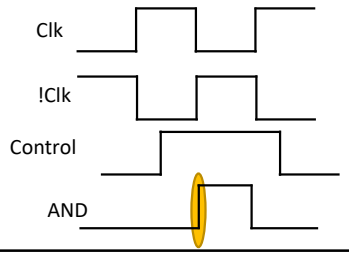


Load PC High (v2.0)

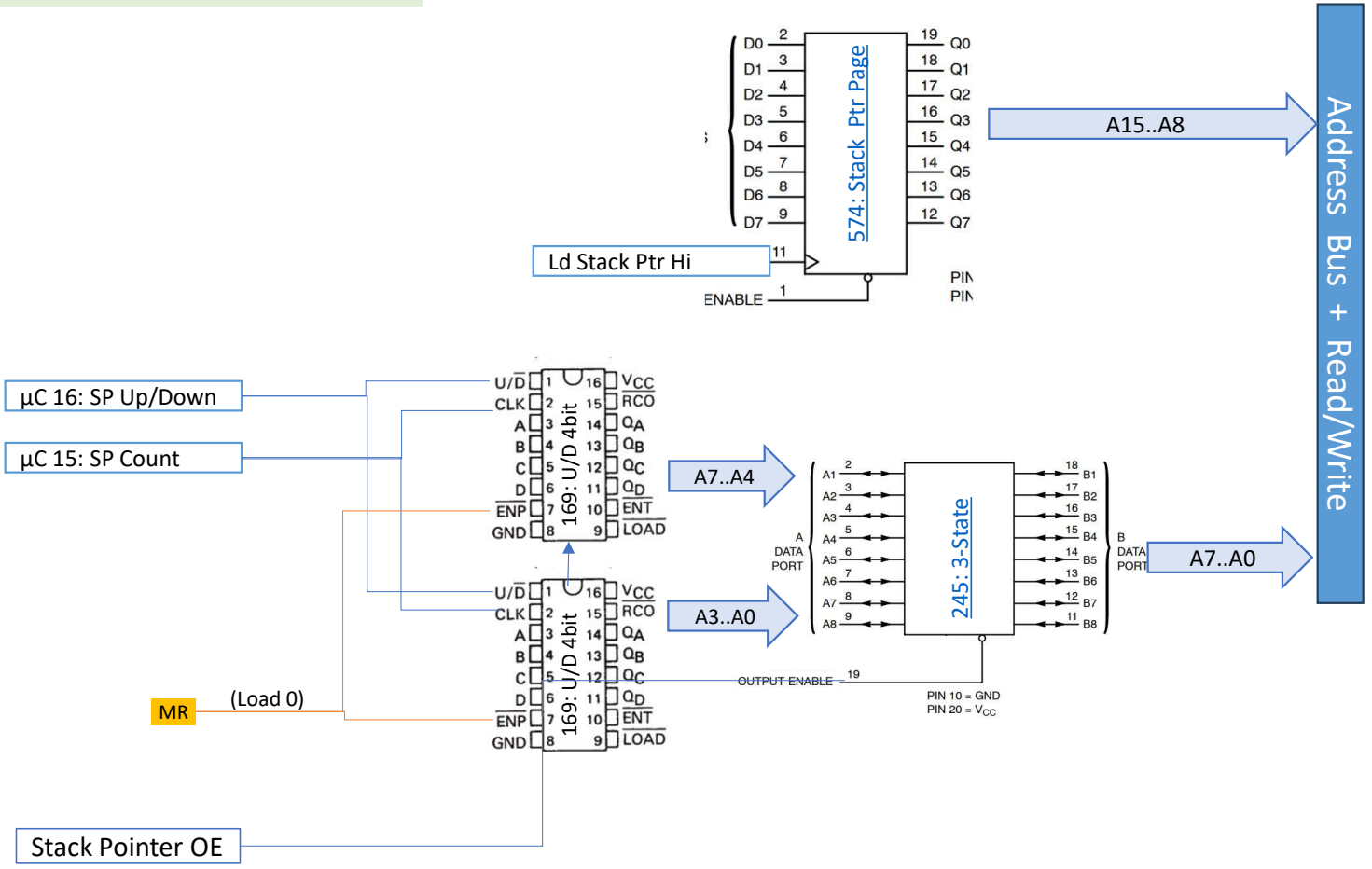


Data Pointer (v2.0)

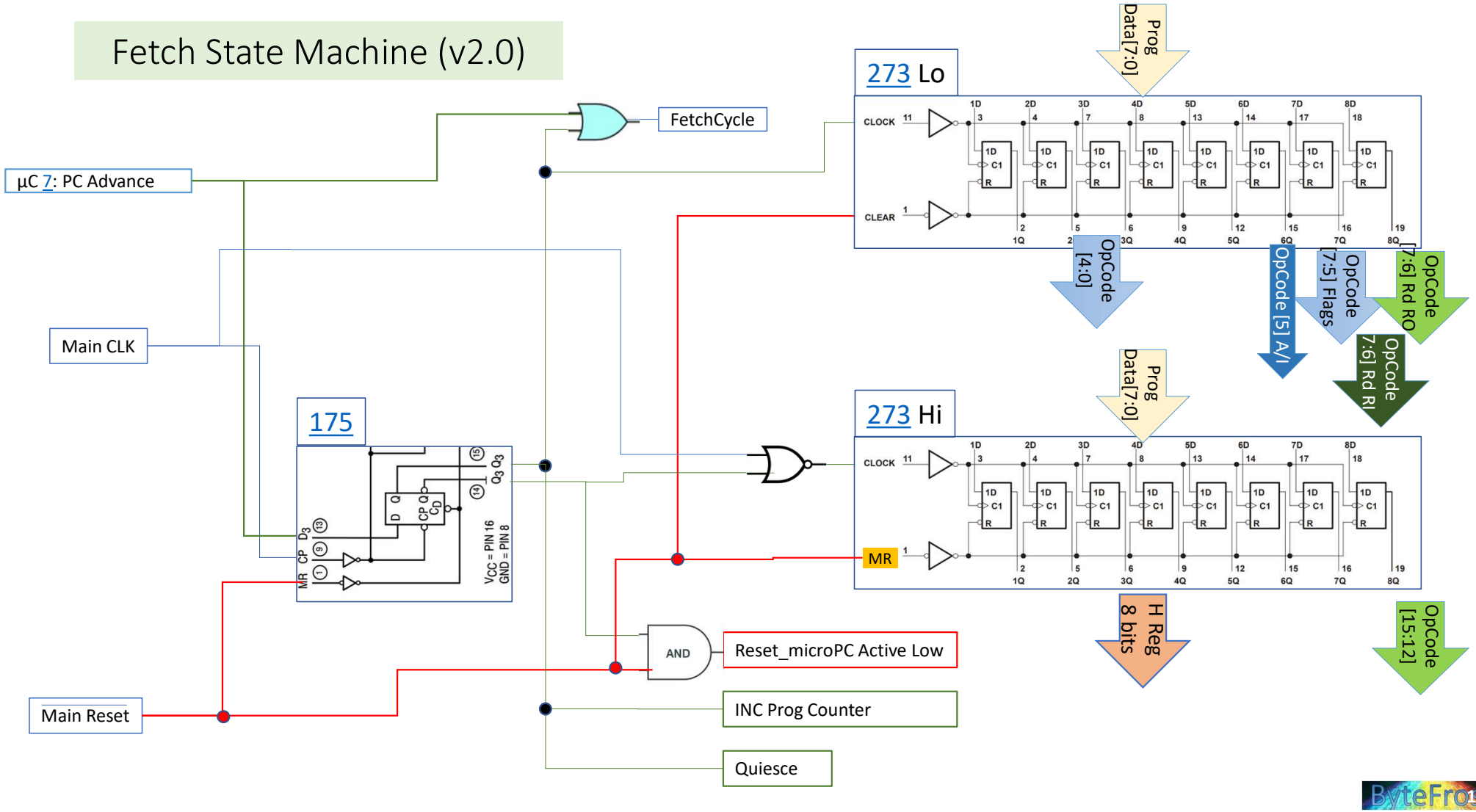




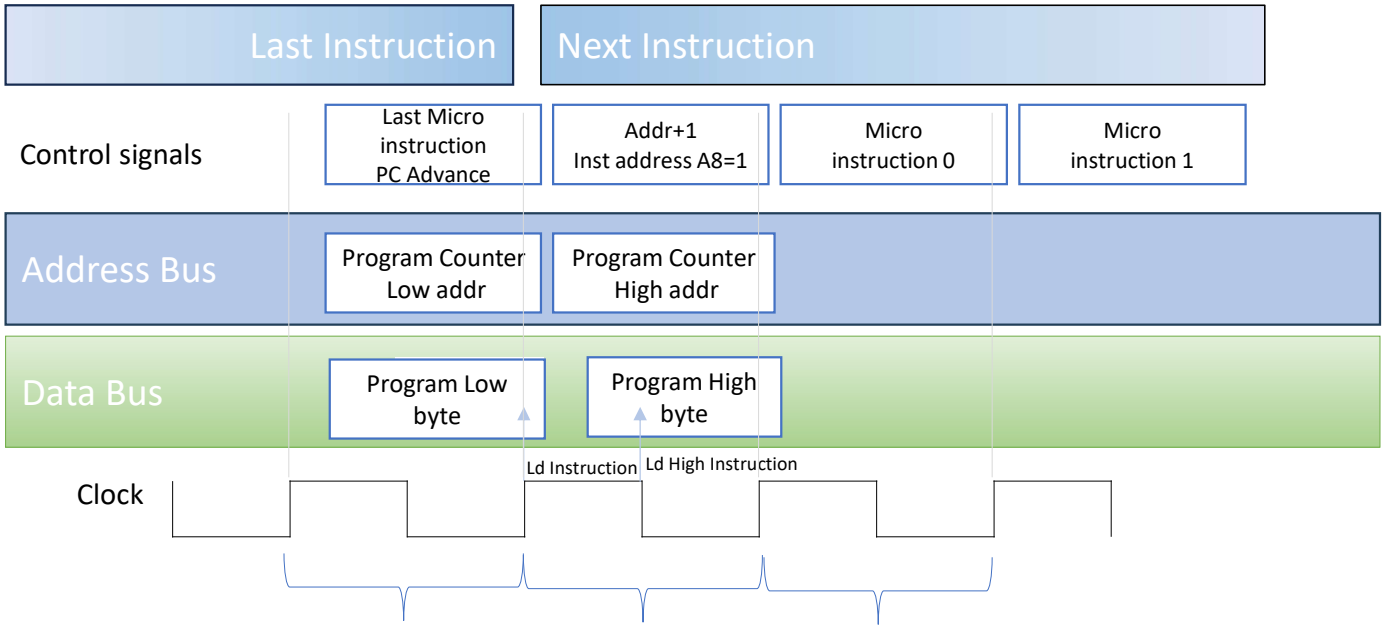
Stack Pointer (v2.0)



Fetch State Machine (v2.0)



Sharing the Data and Address Buses during Fetch Cycle

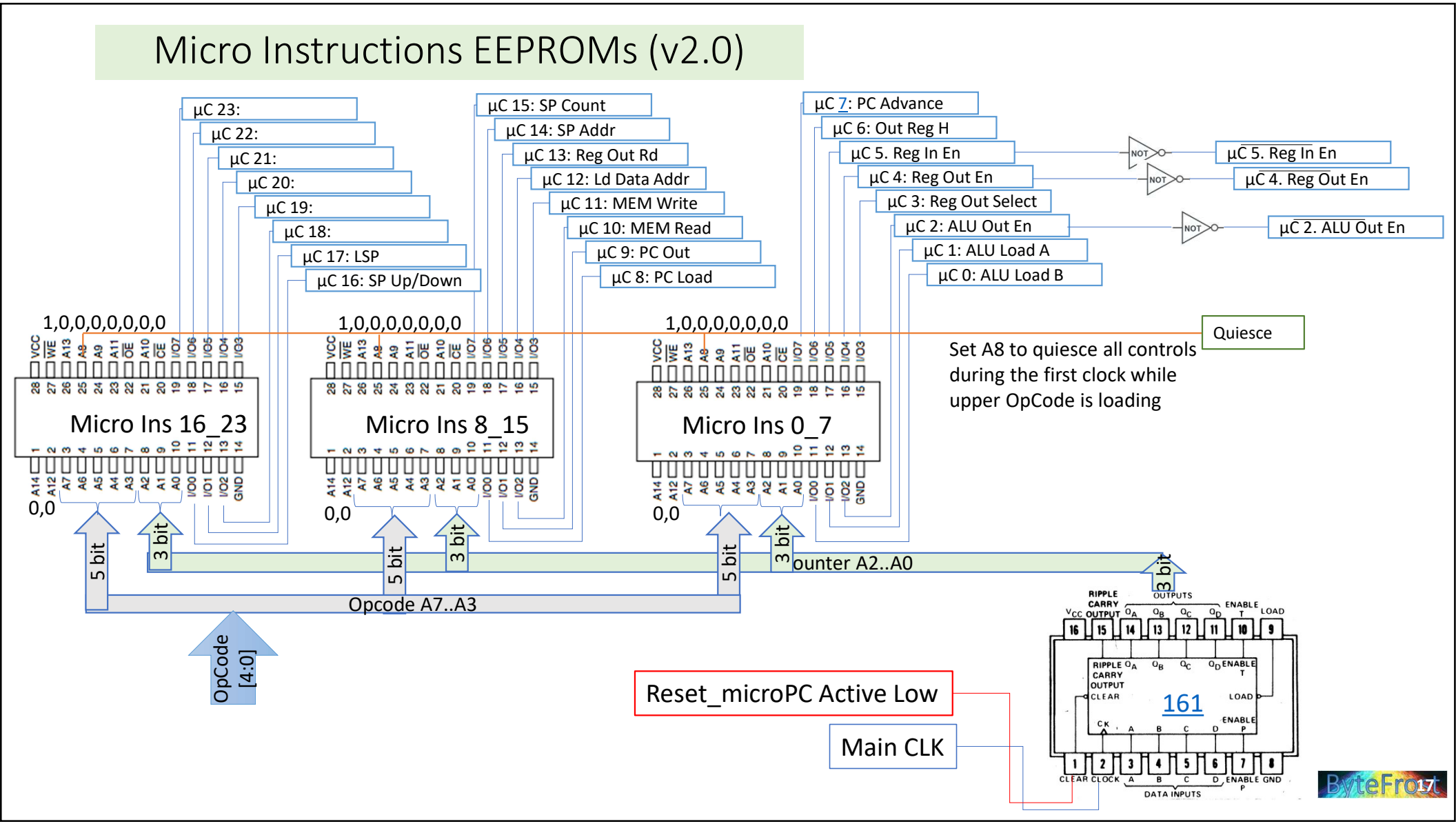


Last clock of an instruction includes the PC_Advance. In that clock, the Address and Data buses are used by the Program Counter. No other signal using the address or data buses is allowed.

First clock of the next instruction, buses are used to load the high byte of the instruction. To avoid transmission of the first micro instruction on the buses, the micro instruction address bit A8 is set, and all control signals are set to 0, avoiding any transmission on the busses

Micro Instruction 0 controls the buses

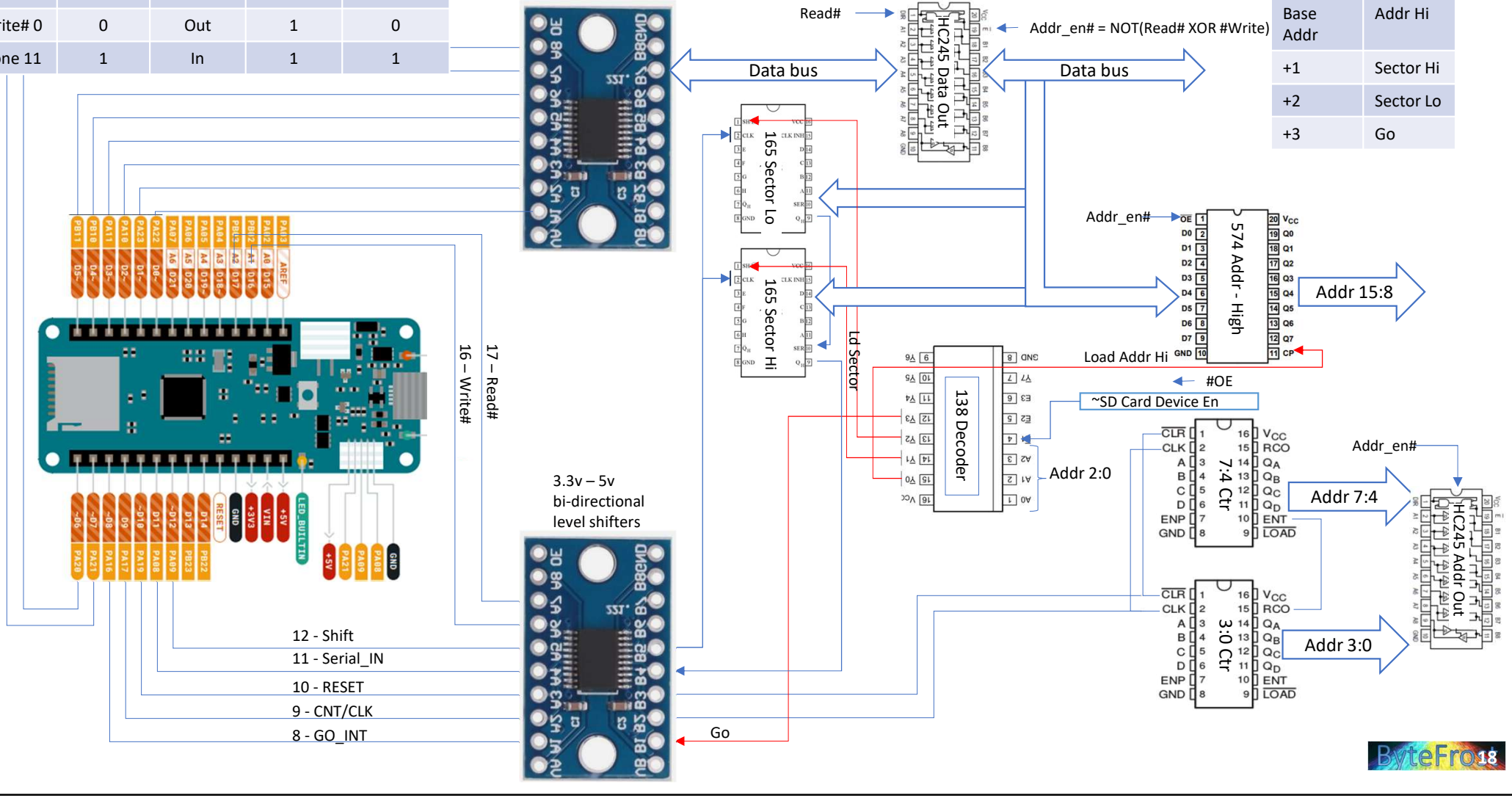
Micro Instructions EEPROMs (v2.0)



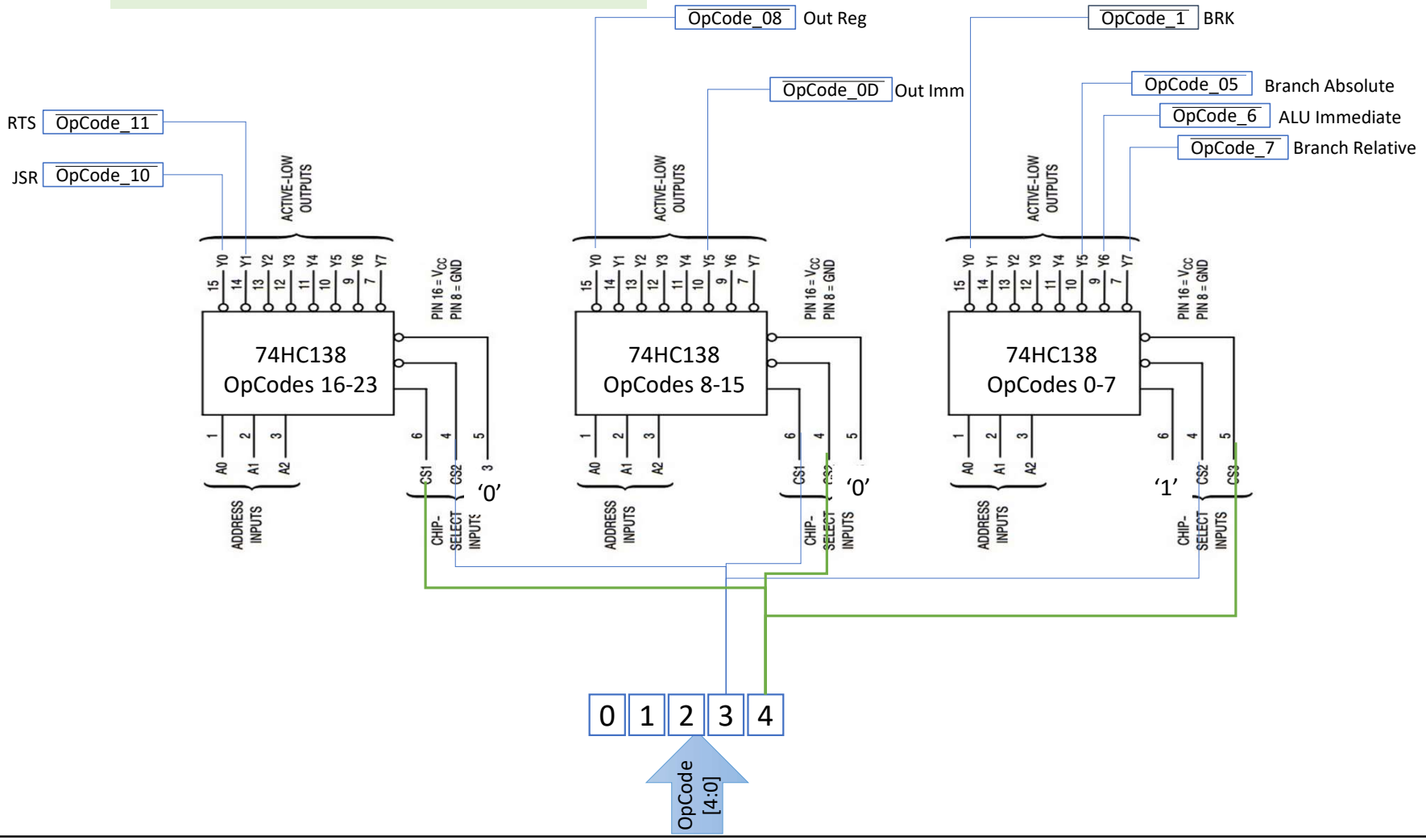
Operation	Addr_En #	Dat_ dir	SRAM OE#	SRAM WE#
Read# 0	0	In	0	1
Write# 0	0	Out	1	0
None 11	1	In	1	1

SD Card – Page Load / Store

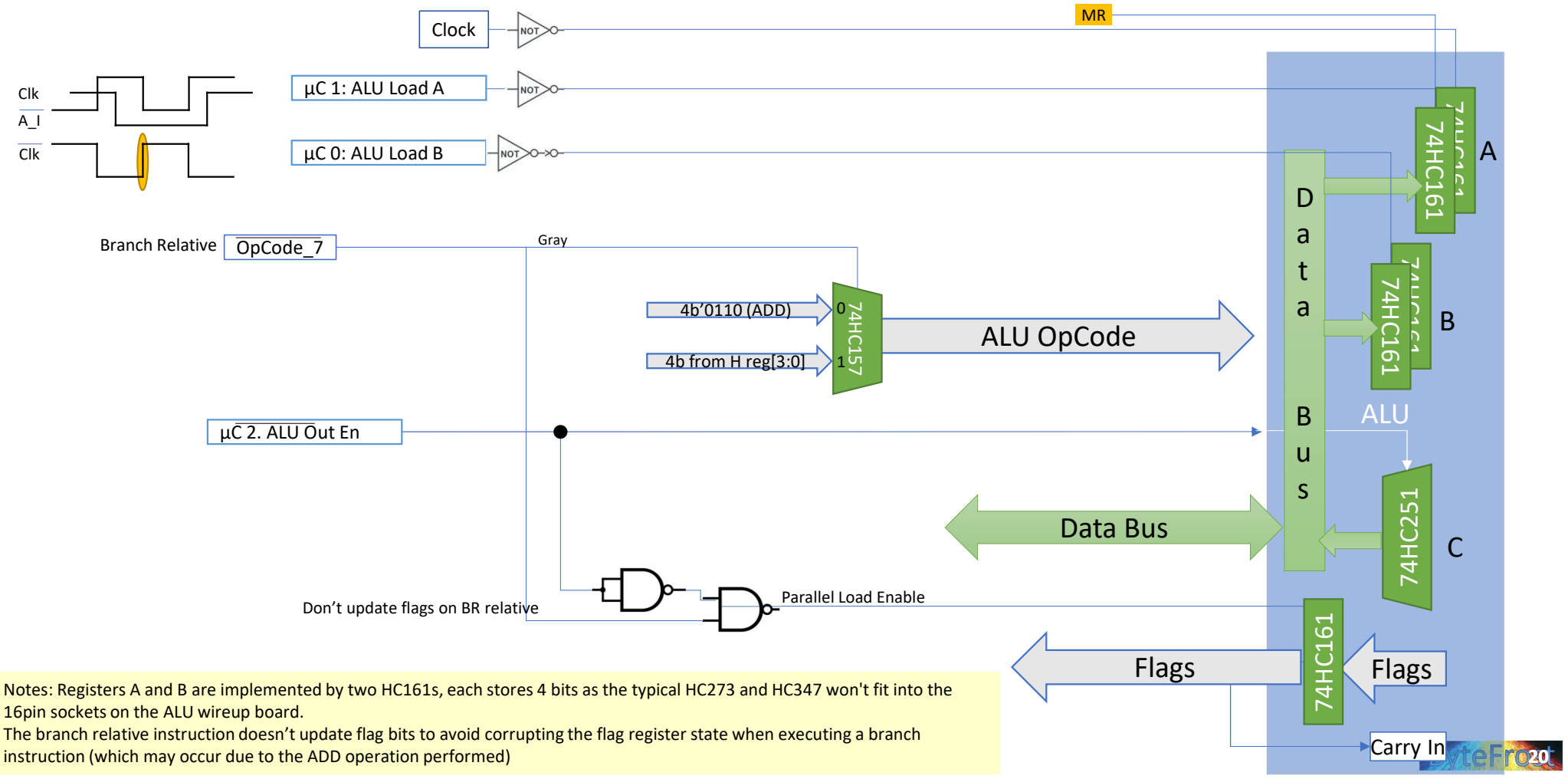
Address	Register
Base Addr	Addr Hi
+1	Sector Hi
+2	Sector Lo
+3	Go



OpCode Decoder (v2.0)

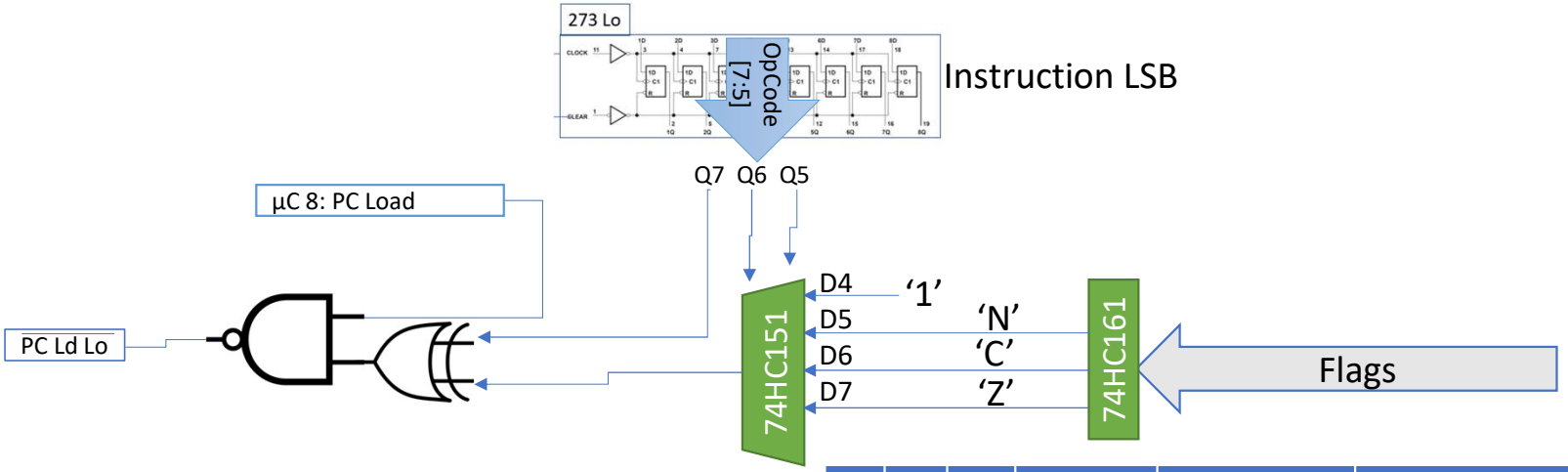


ALU Interface – Flags, A,B Registers, Op Code selection (v2.0)



Notes: Registers A and B are implemented by two HC161s, each stores 4 bits as the typical HC273 and HC347 won't fit into the 16pin sockets on the ALU wireup board.
The branch relative instruction doesn't update flag bits to avoid corrupting the flag register state when executing a branch instruction (which may occur due to the ADD operation performed)

Branch (v2.0)



C2	C1	C0	Condition	Instruction LSB	Equivalent Assembly Instruction
0	0	0	JMP	05	JMP
0	0	1	N	25	BMI – Branch on Minus
0	1	0	C	45	BCS – Branch Carry Set
0	1	1	Z	65	BEQ – Branch Equal
1	0	0	Never	85	NOP
1	0	1	!N	A5	BPL – Branch on Plus
1	1	0	!C	C5	BCC – Branch Carry Clear
1	1	1	!Z	E5	BNE

Instruction set (v2.0)

Op code	Instruction	Details
0x00	NOP	No operation
0x01	BRK	Halt the program
0x02	ALU (Format ALU Rd, Rs1, Rs2)	ALU instruction (OR, AND, XOR, NOT, ADD, SUB, ASL, ROL, LSR, ASR, ROR, ADC, SBC)
0x03	LDR	Load Register from immediate
0x04	MOV	Copy Register (MOV instruction in ARM)
0x05	Branch Absolute	Jump to an address (8bit)
0x06	ALU Immediate (Format: ALU Rd, Rd, imm)	ALU instruction with value from immediate
0x07	BRANCH Relative	Relative Branch (+127 to -128)
0x08	OUT register	Print a register as a character or integer
0x09	LMA	Load register from address
0x0A	SMA	Store register to address
0x0B	LMR	Load register from address in another register
0x0C	SMR	Store register into an address in another register
0x0D	OUT Immediate	Print the immediate next byte as a character or integer
0x0E	PUSH	
0x0F	POP	
0x10	JSR	Jump Subroutine, store current PC in stack
0x11	RTS	Return Subroutine, Load PC from stack
0x12	TST Registers	Compare registers
0x13	TST Register	Compare register and immediate
0x14	LD Special Pointer	Data Pointer. Stack Pointer, Program counter
0x15	Store Special Pointer	

Covered by BIST

Completed

Planned

ALU with changes

Op Select	Special Func. = 0		Special Func. = 1		Flag in	Flags Out
0	A OR B	OR: 0	A OR B		-	Z, N
1	A AND B	AND: 2	A AND B		-	
2	A XOR B	XOR: 4	~B	NOT: 5	-	Z, N
3	A + B	ADD: 6	A – B	SUB: 7	-	Z, N, Co
4	Shift Left A A[0]=0	ASL: 8	Rotate left A A[0]=Cin	ROL: 9	Cin (Rotate)	Z, N=A[6], Co=A[7]
5	Shift Right A	LSR: A	Arithmetic S.R. A	ASR: B	-	Z, N, Co=A[0]
6	Rotate Right	ROR: C	Cin -> A-> Cout		Cin	Z, N, Co=A[0]
7	A + B + Cin	ADC: E	A – B + Cin	SBC: F	Cin	Z, N, Co

Proposal	# of IC slots	Description	Op Code
OR and ADD share Op Code (low priority)	2x 74LS157	Special Function select OR or AND	Release op code
Allow using ALU as two registers (low Priority)		Need to create output A and B	

Micro-Instructions Control: NOP Instruction (00000)

0

Bit	Operation
17	Load Special Register
16	Stack Pointer Increment / Decrement (0: decrease / 1: increase)
15	Stack Pointer Count
14	RAM Address Select (0: Address register / 1: Stack pointer)
13	Use Rd as Source
12	Lower Address Register Load
11	Mem Write
10	Mem Read
9	PC Out
8	PC Load
7	PC advance
6	Program Register H Write to Bus (parameter)
5	Register File Input Enable
4	Register File Output Enable
3	Register File Output Select (parameter – 0 -> Rs1, 1 -> Rs2)
2	ALU output enable
1	ALU load register A
0	ALU load register B

Cycle #	Instruction	Explanation
0	00000000 = 0x00	
1	10000000 = 0x80	Increment PC
2		
3		
4		
5		
6		
7		

NOP

Type	15 14	13 12	11 10 9 8	7 6	5	4 3 2 1 0
None	XX	X	XXXX	XX	X	OpCode=00

Micro-Instructions Control: BREAK (00001)

1

Bit	Operation
17	Load Special Register
16	Stack Pointer Increment / Decrement (0: decrease / 1: increase)
15	Stack Pointer Count
14	RAM Address Select (0: Address register / 1: Stack pointer)
13	Use Rd as Source
12	Lower Address Register Load
11	Mem Write
10	Mem Read
9	PC Out
8	PC Load
7	PC advance
6	Program Register H Write to Bus (parameter)
5	Register File Input Enable
4	Register File Output Enable
3	Register File Output Select (parameter – 0 -> Rs1, 1 -> Rs2)
2	ALU output enable
1	ALU load register A
0	ALU load register B

Cycle #	Instruction	Explanation
0	00000000 = 0x00	Do nothing
1	10000000 = 0x80	Increment PC
2		
3		
4		
5		
6		
7		

Note: BRK has the same microcode as NOP, but the opcode decoder halts the clock when the opcode 00001 is detected. Stepping through it manually (via Single-Step / Run button press) resumes the clock. Hence, BRK acts as a hardware breakpoint.

BRK

Type	15 14	13 12	11 10 9 8	7 6	5	4 3 2 1 0
None	XX	X	XXXX	XX	X	OpCode=01

Micro-Instructions Control: ALU (00010)

2

Bit	Operation
17	Load Special Register
16	Stack Pointer Increment / Decrement (0: decrease / 1: increase)
15	Stack Pointer Count
14	RAM Address Select (0: Address register / 1: Stack pointer)
13	Use Rd as Source
12	Lower Address Register Load
11	Mem Write
10	Mem Read
9	PC Out
8	PC Load
7	PC advance
6	Program Register H Write to Bus (parameter)
5	Register File Input Enable
4	Register File Output Enable
3	Register File Output Select (parameter – 0 -> Rs1, 1 -> Rs2)
2	ALU output enable
1	ALU load register A
0	ALU load register B

Cycle #	Instruction	Explanation
0	00010010 = 0x12	Load Rs1 to reg A
1	00011001 = 0x19	Load Rs2 to reg B
2	00100100 = 0x24	Output and Read to Rd
3	10000000 = 0x80	Increment PC
4		
5		
6		
7		

```
ADD Rd, Rs1, Rs2
```

Type	15 14	13 12	11 10 9 8	7 6	5	4 3 2 1 0
ALU	Rs2	Rs1	S2 S1 S0 SF (Function sel)	Rd	X	OpCode=02

Micro-Instructions Control: Load Immediate to Reg (00011)

3

Bit	Operation
17	Load Special Register
16	Stack Pointer Increment / Decrement (0: decrease / 1: increase)
15	Stack Pointer Count
14	RAM Address Select (0: Address register / 1: Stack pointer)
13	Use Rd as Source
12	Lower Address Register Load
11	Mem Write
10	Mem Read
9	PC Out
8	PC Load
7	PC advance
6	Program Register H Write to Bus (parameter)
5	Register File Input Enable
4	Register File Output Enable
3	Register File Output Select (parameter – 0 -> Rs1, 1 -> Rs2)
2	ALU output enable
1	ALU load register A
0	ALU load register B

Cycle #	Instruction	Explanation
0	01100000 = 0x60	Write imm to bus and Rd read
1	10000000 = 0x80	Increment PC
2		
3		
4		
5		
6		
7		

```
LDR Rd, #0x00
```

Type	15 14	13 12	11 10 9 8	7 6	5	4 3 2 1 0
Load Imm	Immediate (8-bit)			Rd	X	OpCode=03

Micro-Instructions Control: MOV (00100)

4

Bit	Operation
17	Load Special Register
16	Stack Pointer Increment / Decrement (0: decrease / 1: increase)
15	Stack Pointer Count
14	RAM Address Select (0: Address register / 1: Stack pointer)
13	Use Rd as Source
12	Lower Address Register Load
11	Mem Write
10	Mem Read
9	PC Out
8	PC Load
7	PC advance
6	Program Register H Write to Bus (parameter)
5	Register File Input Enable
4	Register File Output Enable
3	Register File Output Select (parameter – 0 -> Rs1, 1 -> Rs2)
2	ALU output enable
1	ALU load register A
0	ALU load register B

Cycle #	Instruction	Explanation
0	00110000 = 0x30	Write Rs1 to bus and Rd reads
1	10000000 = 0x80	Increment PC
2		
3		
4		
5		
6		
7		

```
MOV Rd, Rs1
```

Type	15 14	13 12	11 10 9 8	7 6	5	4 3 2 1 0
Load Reg	XX	Rs1	XXXX	Rd	X	OpCode=04

Micro-Instructions Control: Branch / Jump Absolute Immediate (00101)

5

Bit	Operation
17	Load Special Register
16	Stack Pointer Increment / Decrement (0: decrease / 1: increase)
15	Stack Pointer Count
14	RAM Address Select (0: Address register / 1: Stack pointer)
13	Use Rd as Source
12	Lower Address Register Load
11	Mem Write
10	Mem Read
9	PC Out
8	PC Load
7	PC advance
6	Program Register H Write to Bus (parameter)
5	Register File Input Enable
4	Register File Output Enable
3	Register File Output Select (parameter – 0 -> Rs1, 1 -> Rs2)
2	ALU output enable
1	ALU load register A
0	ALU load register B

Cycle #	Instruction	Explanation
0	0x1_40 01_0100_0000	Write H register to bus and PC (conditionally) reads
1	0x0_80 00_1000_0000	PC advance
2		
3		
4		
5		
6		
7		

C2	C1	C0	Condition	Low instruction register	Assembly
0	0	0	JMP	05	JMP
0	0	1	N	25	BMI – Branch on Minus
0	1	0	C	45	BCS – Branch Carry Set
0	1	1	Z	65	BEQ – Branch Equal
1	0	0	Never	85	NOP
1	0	1	!N	A5	BPL – Branch on Plus
1	1	0	!C	C5	BCC – Branch Carry Clear
1	1	1	!Z	E5	BNE

Type	15 14	13 12	11 10 9 8	7 6 5	4 3 2 1 0
Branch	Immediate (8-bit)			C2 C1 C0	OpCode=05

Micro-Instructions Control: ALU immediate (4-bit) (00110)

Bit	Operation
17	Load Special Register
16	Stack Pointer Increment / Decrement (0: decrease / 1: increase)
15	Stack Pointer Count
14	RAM Address Select (0: Address register / 1: Stack pointer)
13	Use Rd as Source
12	Lower Address Register Load
11	Mem Write
10	Mem Read
9	PC Out
8	PC Load
7	PC advance
6	Program Register H Write to Bus (parameter)
5	Register File Input Enable
4	Register File Output Enable
3	Register File Output Select (parameter – 0 -> Rs1, 1 -> Rs2)
2	ALU output enable
1	ALU load register A
0	ALU load register B

Cycle #	Instruction	Explanation
0	10_0000_0001_0010 = 2012	Load Rd to reg A
1	0100_0001 = 0x0041	Load 4 bit imm to reg B
2	0010_0100 = 0x0024	Write ALU output to Rd
3	1000_0000 = 0x0080	Increment PC
4		
5		
6		
7		

```
ADD Rd, #0x00
```

Type	15 14 13 12	11 10 9 8	7 6	5	4 3 2 1 0
ALU	4-bit imm.	S2 S1 S0 SF (Function sel)	Rd	X	OpCode=06

Micro-Instructions Control: Branch / Jump Relative Immediate (00111)

7

Bit	Operation
17	Load Special Register
16	Stack Pointer Increment / Decrement (0: decrease / 1: increase)
15	Stack Pointer Count
14	RAM Address Select (0: Address register / 1: Stack pointer)
13	Use Rd as Source
12	Lower Address Register Load
11	Mem Write
10	Mem Read
9	PC Out
8	PC Load
7	PC advance
6	Program Register H Write to Bus (parameter)
5	Register File Input Enable
4	Register File Output Enable
3	Register File Output Select (parameter – 0 -> Rs1, 1 -> Rs2)
2	ALU output enable
1	ALU load register A
0	ALU load register B

Cycle #	Instruction	Explanation
0	10_0000_0010 0x2_02	Load PC into reg A
1	00_0100_0001 0x0_41	Load immediate into reg B
2	01_0000_0100 0x1_04	Output ALU val, load PC
3	00_1000_0000 0x0_80	Advance PC
4		
5		
6		
7		

C2	C1	C0	Condition	OpCode	Assembly
0	0	0	JMP	07	JMP
0	0	1	N	27	BMI – Branch on Minus
0	1	0	C	47	BCS – Branch Carry Set
0	1	1	Z	67	BEQ – Branch Equal
1	0	0	Never	87	NOP
1	0	1	!N	A7	BPL – Branch on Plus
1	1	0	!C	C7	BCC – Branch Carry Clear
1	1	1	!Z	E7	BNE

Type	15 14	13 12	11 10 9 8	7 6 5	4 3 2 1 0
Branch	Immediate (8-bit)			C2 C1 C0	OpCode=07

Micro-Instructions Control: OUT (01000) (Display ASCII or Integer)

Bit	Operation
17	Load Special Register
16	Stack Pointer Increment / Decrement (0: decrease / 1: increase)
15	Stack Pointer Count
14	RAM Address Select (0: Address register / 1: Stack pointer)
13	Use Rd as Source
12	Lower Address Register Load
11	Mem Write
10	Mem Read
9	PC Out
8	PC Load
7	PC advance
6	Program Register H Write to Bus (parameter)
5	Register File Input Enable
4	Register File Output Enable
3	Register File Output Select (parameter – 0 -> Rs1, 1 -> Rs2)
2	ALU output enable
1	ALU load register A
0	ALU load register B

Cycle #	Instruction	Explanation
0	0000_0000 = 0x00	Wait for H register to be loaded (need value of Rs1)- No longer required
1	0001_0000 = 0x10	Write Rs1 to the bus (when opcode 01000 is detected, the value on the bus is read by the OUT register to the display)
2	1000_0000 = 0x80	Advance PC
3		
4		

```
OUT Rs1, A
```

Type	15 14	13 12	11 10 9 8	7 6	5	4 3 2 1 0
Out	XX	Rs1	XXXX	XX	I=1; A=0	OpCode=08

Micro-Instructions Control: LMA (Load Memory Absolute) (01001)

Bit	Operation
17	Load Special Register
16	Stack Pointer Increment / Decrement (0: decrease / 1: increase)
15	Stack Pointer Count
14	RAM Address Select (0: Address register / 1: Stack pointer)
13	Use Rd as Source
12	Lower Address Register Load
11	Mem Write
10	Mem Read
9	PC Out
8	PC Load
7	PC advance
6	Program Register H Write to Bus (parameter)
5	Register File Input Enable
4	Register File Output Enable
3	Register File Output Select (parameter – 0 -> Rs1, 1 -> Rs2)
2	ALU output enable
1	ALU load register A
0	ALU load register B

Cycle #	Instruction	Explanation
0	1_0000_0100_0000 = 1040	Load 8-bit address to address register
1	0_0100_0010_0000 = 0420	Read byte from RAM and store in Rd
2	0_0000_1000_0000 = 0080	Increment PC
3		
4		
5		
6		
7		

```
LMA Rd, #Addr
```

Type	15 14 13 12 11 10 9 8	7 6	5	4 3 2 1 0
Load Reg	8-bit (LSBs) Address	Rd	X	OpCode=09

Micro-Instructions Control: SMA (Store Memory Absolute) (01010)

A

Bit	Operation
17	Load Special Register
16	Stack Pointer Increment / Decrement (0: decrease / 1: increase)
15	Stack Pointer Count
14	RAM Address Select (0: Address register / 1: Stack pointer)
13	Use Rd as Source
12	Lower Address Register Load
11	Mem Write
10	Mem Read
9	PC Out
8	PC Load
7	PC advance
6	Program Register H Write to Bus (parameter)
5	Register File Input Enable
4	Register File Output Enable
3	Register File Output Select (parameter – 0 -> Rs1, 1 -> Rs2)
2	ALU output enable
1	ALU load register A
0	ALU load register B

Cycle #	Instruction	Explanation
0	01_0000_0100_0000 = 1040	Load 8-bit address to address register
1	10_1000_0001_0000 = 2810	Store byte to RAM from Rd
2	00_0000_1000_0000 = 0080	Increment PC
3		
4		
5		
6		
7		

```
SMA Rd, #Addr
```

Type	15 14 13 12 11 10 9 8	7 6	5	4 3 2 1 0
Load Reg	8-bit (LSBs) Address	Rd	X	OpCode=0A

Micro-Instructions Control: LMR (Load Memory Register) (01011)

B

Bit	Operation
17	Load Special Register
16	Stack Pointer Increment / Decrement (0: decrease / 1: increase)
15	Stack Pointer Count
14	RAM Address Select (0: Address register / 1: Stack pointer)
13	Use Rd as Source
12	Lower Address Register Load
11	Mem Write
10	Mem Read
9	PC Out
8	PC Load
7	PC advance
6	Program Register H Write to Bus (parameter)
5	Register File Input Enable
4	Register File Output Enable
3	Register File Output Select (parameter – 0 -> Rs1, 1 -> Rs2)
2	ALU output enable
1	ALU load register A
0	ALU load register B

Cycle #	Instruction	Explanation
0	1_0000_0001_0000 = 1010	Load Rs1 to Address Register
1	0_0100_0010_0000 = 0420	Load value at the address into Rd
2	0_0000_1000_0000 = 0080	Increment PC
3		
4		
5		
6		
7		

```
LMR Rd, Rs
```

Effect: Rd = *Rs

Type	15 14	13 12	11 10 9 8	7 6	5	4 3 2 1 0
Load Reg	XX	Rs1	XXXX	Rd	X	OpCode=0B

Micro-Instructions Control: SMR (Store Memory Register) (01100)

Bit	Operation
17	Load Special Register
16	Stack Pointer Increment / Decrement (0: decrease / 1: increase)
15	Stack Pointer Count
14	RAM Address Select (0: Address register / 1: Stack pointer)
13	Use Rd as Source
12	Lower Address Register Load
11	Mem Write
10	Mem Read
9	PC Out
8	PC Load
7	PC advance
6	Program Register H Write to Bus (parameter)
5	Register File Input Enable
4	Register File Output Enable
3	Register File Output Select (parameter – 0 -> Rs1, 1 -> Rs2)
2	ALU output enable
1	ALU load register A
0	ALU load register B

Cycle #	Instruction	Explanation
0	01_0000_0001_0000 = 1010	Load Rs1 to Address Register
1	10_1000_0001_0000 = 2810	Store value at Rd to RAM
2	00_0000_1000_0000 = 0080	Increment PC
3		
4		
5		
6		
7		

```
SMR Rd_data, Rs1_addr
```

Type	15 14	13 12	11 10 9 8	7 6	5	4 3 2 1 0
Load Reg	XX	Rs1	XXXX	Rd	X	OpCode=0C

Micro-Instructions Control: OUT immediate (01101)

D

Bit	Operation
17	Load Special Register
16	Stack Pointer Increment / Decrement (0: decrease / 1: increase)
15	Stack Pointer Count
14	RAM Address Select (0: Address register / 1: Stack pointer)
13	Use Rd as Source
12	Lower Address Register Load
11	Mem Write
10	Mem Read
9	PC Out
8	PC Load
7	PC advance
6	Program Register H Write to Bus (parameter)
5	Register File Input Enable
4	Register File Output Enable
3	Register File Output Select (parameter – 0 -> Rs1, 1 -> Rs2)
2	ALU output enable
1	ALU load register A
0	ALU load register B

Cycle #	Instruction	Explanation
0	0000_0000 = 0x00	Wait for H instruction register to be loaded
1	0100_0000 = 0x40	Write program register H to the bus
2	1000_0000 = 0x80	Advance PC
3		
4		

```
OUT #Imm, A
```

Type	15 14	13 12	11 10 9 8	7 6	5	4 3 2 1 0
Out	8-bit immediate			XX	I=1; A=0	OpCode=0D

Micro-Instructions Control: PUSH (01110)

E

Bit	Operation
17	Load Special Register
16	Stack Pointer Increment / Decrement (0: decrease / 1: increase)
15	Stack Pointer Count
14	RAM Address Select (0: Address register / 1: Stack pointer)
13	Use Rd as Source
12	Lower Address Register Load
11	Mem Write
10	Mem Read
9	PC Out
8	PC Load
7	PC advance
6	Program Register H Write to Bus (parameter)
5	Register File Input Enable
4	Register File Output Enable
3	Register File Output Select (parameter – 0 -> Rs1, 1 -> Rs2)
2	ALU output enable
1	ALU load register A
0	ALU load register B

Cycle #	Instruction	Explanation
0	0_0100_1000_0001_0000 = 0x04810	Store Rs1 in the stack
1	1_1000_0000_1000_0000 = 0x18080	Increment stack pointer (and increment PC)
2		
3		
4		
5		
6		

PUSH Rs1

Type	15 14	13 12	11 10 9 8	7 6	5	4 3 2 1 0
Load Reg	XX	Rs1	XXXX	XX	X	OpCode=0E

Micro-Instructions Control: POP (1111)

F

Bit	Operation
17	Load Special Register
16	Stack Pointer Increment / Decrement (0: decrease / 1: increase)
15	Stack Pointer Count
14	RAM Address Select (0: Address register / 1: Stack pointer)
13	Use Rd as Source
12	Lower Address Register Load
11	Mem Write
10	Mem Read
9	PC Out
8	PC Load
7	PC advance
6	Program Register H Write to Bus (parameter)
5	Register File Input Enable
4	Register File Output Enable
3	Register File Output Select (parameter – 0 -> Rs1, 1 -> Rs2)
2	ALU output enable
1	ALU load register A
0	ALU load register B

Cycle #	Instruction	Explanation
0	0_0000_0000_0000_0000 = 0x00000	Wait for next clock
1	0_1000_0000_0000_0000 = 0x08000	Decrement stack pointer
2	0_0100_0100_0010_0000 = 0x04420	Load value from stack to Rd
3	0_0000_0000_1000_0000 = 0x00080	Increment PC
4		
5		
6		

POP Rd

Type	15 14	13 12	11 10 9 8	7 6	5	4 3 2 1 0
Load Reg	XX	XX	XXXX	Rd	X	OpCode=0F

Micro-Instructions Control: JSR Jump Subroutine (10000)

Bit	Operation
17	Load Special Register
16	Stack Pointer Increment / Decrement (0: decrease / 1: increase)
15	Stack Pointer Count
14	RAM Address Select (0: Address register / 1: Stack pointer)
13	Use Rd as Source
12	Lower Address Register Load
11	Mem Write
10	Mem Read
9	PC Out
8	PC Load
7	PC advance
6	Program Register H Write to Bus (parameter)
5	Register File Input Enable
4	Register File Output Enable
3	Register File Output Select (parameter – 0 -> Rs1, 1 -> Rs2)
2	ALU output enable
1	ALU load register A
0	ALU load register B

Cycle #	Instruction	Explanation
0	0_0100_1010_0000_0000 = 0x04A00	Write PC to stack
1	1_1000_0000_0000_0000 = 0x18000	Increase stack pointer
2	0_0000_0001_0100_0000 = 0x00140	Jump to immediate address
3	0_0000_0000_1000_0000 = 0x00080	Advance PC
4		
5		
6		

JSR #Imm

Type	15 14 13 12 11 10 9 8	7 6	5	4 3 2 1 0
Load Reg	8-bit Immediate (Program Address)	XX	X	OpCode=10

Micro-Instructions Control: RTS Return from Subroutine (10001)

11

Bit	Operation
17	Load Special Register
16	Stack Pointer Increment / Decrement (0: decrease / 1: increase)
15	Stack Pointer Count
14	RAM Address Select (0: Address register / 1: Stack pointer)
13	Use Rd as Source
12	Lower Address Register Load
11	Mem Write
10	Mem Read
9	PC Out
8	PC Load
7	PC advance
6	Program Register H Write to Bus (parameter)
5	Register File Input Enable
4	Register File Output Enable
3	Register File Output Select (parameter – 0 -> Rs1, 1 -> Rs2)
2	ALU output enable
1	ALU load register A
0	ALU load register B

Cycle #	Instruction	Explanation
0	00_1000_0000_0000_0000 = 0x08000	Decrement stack pointer
1	00_1000_0000_0000_0000 = 0x08000	Decrement stack pointer
2	10_0100_0100_0000_0000 = 0x24400	Load Dummy from stack
3	01_1000_0000_0000_0000 = 0x18000	Increment SP
4	00_0100_0101_0000_0000 = 0x04500	Load program address on the stack to the program counter
5	00_1000_0000_1000_0000 = 0x08080	Decrement SP, Advance PC

Stack Pos	Expected Data
Start SP	Empty
Start SP - 1	Return Address Low Byte
Start SP – 2	Return Address High Byte

Note: Return address must be 16-bit! I.e., to jump to a subroutine, you must use:
PUSH Return Address High Byte
LSP DummyPC, Subroutine Address High Byte
JSR Subroutine Address Low Byte

RTS

Type	15 14 13 12 11 10 9 8	7 6	5	4 3 2 1 0
Load Reg	XXXX XXXX	00	X	OpCode=11

Micro-Instructions Control: Test (10010)

Bit	Operation
17	Load Special Register
16	Stack Pointer Increment / Decrement (0: decrease / 1: increase)
15	Stack Pointer Count
14	RAM Address Select (0: Address register / 1: Stack pointer)
13	Use Rd as Source
12	Lower Address Register Load
11	Mem Write
10	Mem Read
9	PC Out
8	PC Load
7	PC advance
6	Program Register H Write to Bus (parameter)
5	Register File Input Enable
4	Register File Output Enable
3	Register File Output Select (parameter – 0 -> Rs1, 1 -> Rs2)
2	ALU output enable
1	ALU load register A
0	ALU load register B

Cycle #	Instruction	Explanation
0	0001_0010 = 0x12	Load Rs1 to reg A
1	0001_1001 = 0x19	Load Rs2 to reg B
2	0000_0100 = 0x04	Output ALU value on bus (but don't save in register)
3	1000_0000 = 0x80	Increment PC
4		
5		
6		
7		

```
TST Rs1, Rs2
```

Type	15 14	13 12	11 10 9 8	7 6	5	4 3 2 1 0
ALU	Rs2	Rs1	S2 S1 S0 SF (Function sel)	XX	X	OpCode=12

Micro-Instructions Control: Test immediate (10011)

Bit	Operation
17	Load Special Register
16	Stack Pointer Increment / Decrement (0: decrease / 1: increase)
15	Stack Pointer Count
14	RAM Address Select (0: Address register / 1: Stack pointer)
13	Use Rd as Source
12	Lower Address Register Load
11	Mem Write
10	Mem Read
9	PC Out
8	PC Load
7	PC advance
6	Program Register H Write to Bus (parameter)
5	Register File Input Enable
4	Register File Output Enable
3	Register File Output Select (parameter – 0 -> Rs1, 1 -> Rs2)
2	ALU output enable
1	ALU load register A
0	ALU load register B

Cycle #	Instruction	Explanation
0	10_0000_0001_0010 = 0x2012	Load Rd to reg A
1	0100_0001 = 0x0041	Load 4 bit imm to reg B
2	0000_0100 = 0x0004	Write ALU output to bus (but not to register)
3	1000_0000 = 0x80	Increment PC
4		
5		
6		
7		

```
TST Rd, #0x00
```

Type	15 14 13 12	11 10 9 8	7 6	5	4 3 2 1 0
ALU	4-bit imm.	S2 S1 S0 SF (Function sel)	Rd	X	OpCode=13

Micro-Instructions Control: Load Special Register (Immediate) (10100)

Bit	Operation
17	Load Special Register
16	Stack Pointer Increment / Decrement (0: decrease / 1: increase)
15	Stack Pointer Count
14	RAM Address Select (0: Address register / 1: Stack pointer)
13	Use Rd as Source
12	Lower Address Register Load
11	Mem Write
10	Mem Read
9	PC Out
8	PC Load
7	PC advance
6	Program Register H Write to Bus (parameter)
5	Register File Input Enable
4	Register File Output Enable
3	Register File Output Select (parameter – 0 -> Rs1, 1 -> Rs2)
2	ALU output enable
1	ALU load register A
0	ALU load register B

Cycle #	Instruction	Explanation
0	10_0000_0000_0100_0000 = 0x20040	Loads 8-bit immediate to specified special register
1	00_0000_0000_1000_0000 = 0x00080	Increment PC
2		
3		

C1	C0	Special Register	Command Parameter
0	0	PC (High Byte)	PC
0	1	Data Pointer (High Byte)	DP
1	0	Stack Pointer (High Byte)	SP
1	1	N/A (in future, perhaps a Frame Pointer (High Byte))	

```
LSP DP, #0x00
```

Type	15 14	13 12	11 10 9 8	7 6	5	4 3 2 1 0
Branch	Immediate (8-bit)			C1 C0	X	OpCode=14

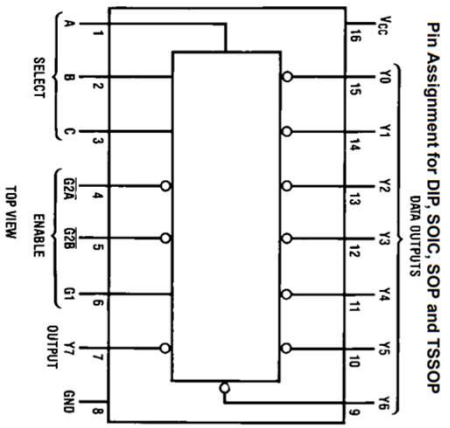
138 - 3-to-8 Line Decoder

Truth Table

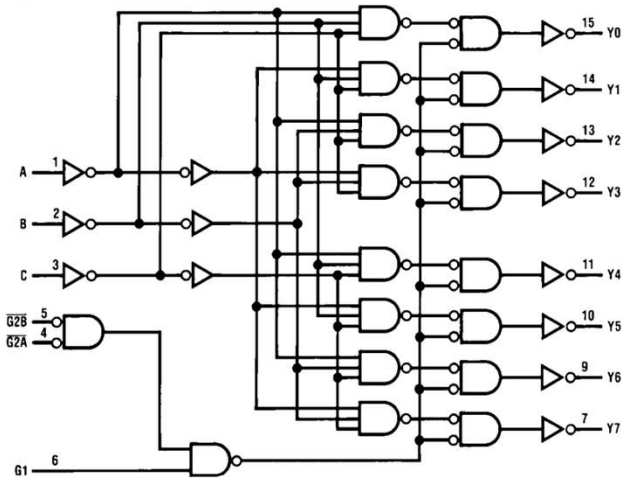
Inputs					Outputs							
Enable		Select										
G1	$\overline{G2}$ (Note 1)	C	B	A	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
X	H	X	X	X	H	H	H	H	H	H	H	H
L	X	X	X	X	H	H	H	H	H	H	H	H
H	L	L	L	L	L	H	H	H	H	H	H	H
H	L	L	L	H	H	L	H	H	H	H	H	H
H	L	L	H	L	H	H	L	H	H	H	H	H
H	L	L	H	H	H	H	H	L	H	H	H	H
H	L	H	L	L	H	H	H	H	L	H	H	H
H	L	H	L	H	H	H	H	H	H	L	H	H
H	L	H	H	L	H	H	H	H	H	H	L	H
H	L	H	H	H	H	H	H	H	H	H	H	L

H = HIGH Level, L = LOW Level, X = don't care

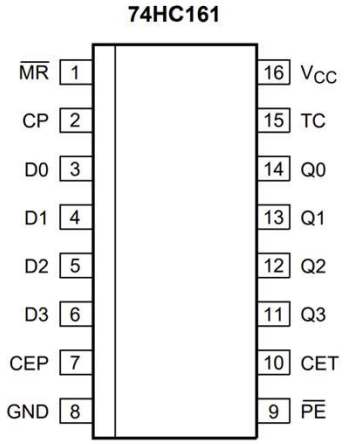
Note 1: $\overline{G2}$ = G2A+G2B



Logic Diagram

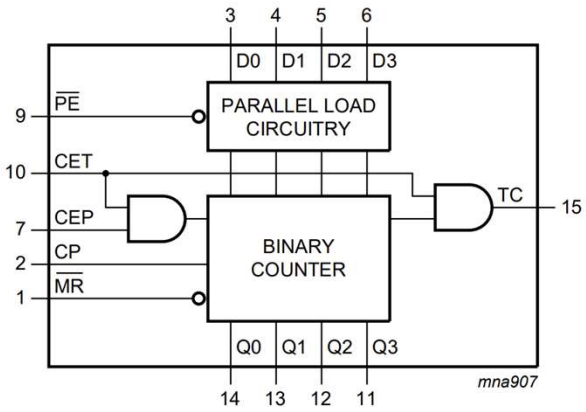


161 - 4-Bit Binary Counter, Asynchronous Reset



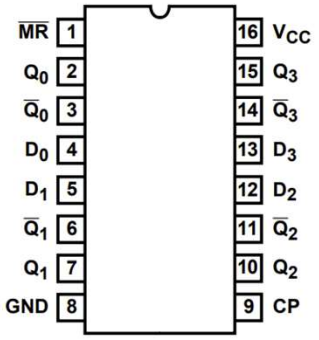
Operating modes	Input						Output	
	$\overline{MR}$	CP	CEP	CET	$\overline{PE}$	Dn	Qn	TC
Reset (clear)	L	X	X	X	X	X	L	L
Parallel load	H	↑	X	X	l	l	L	L
	H	↑	X	X	l	h	H	[1]
Count	H	↑	h	h	h	X	count	[1]
Hold (do nothing)	H	X	l	X	h	X	q_n	[1]
	H	X	X	l	h	X	q_n	L

[1] The TC output is HIGH when CET is HIGH and the counter is at terminal count (HHHH)



Notes:
- Parallel Load need to happen on invese clock
- Count need to happen on [TBD – need timing diagram of the PC Advance]

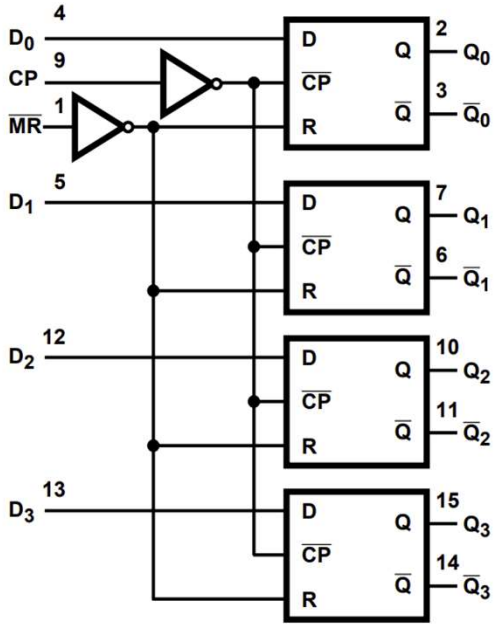
175 - Quad D-Type Flip-Flop with Reset



TRUTH TABLE

INPUTS			OUTPUTS	
RESET ($\overline{MR}$)	CLOCK CP	DATA D_n	Q_n	$\overline{Q}_n$
L	X	X	L	H
H	↑	H	H	L
H	↑	L	L	H
H	L	X	Q_0	$\overline{Q}_0$

H = High Voltage Level, L = Low Voltage Level, X = Don't Care, ↑ = Transition from Low to High Level,
Q_0 = Level Before the Indicated Steady-State Input Conditions Were Established.



245 - Octal D-Type Flip-Flop with Reset

FUNCTION TABLE

Control Inputs		Operation
Output Enable	Direction	
L	L	Data Transmitted from Bus B to Bus A
L	H	Data Transmitted from Bus A to Bus B
H	X	Buses Isolated (High–Impedance State)

X = don't care

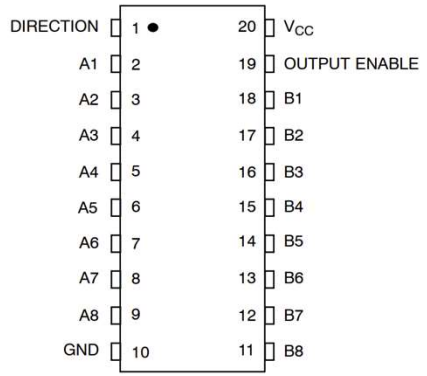


Figure 1. Pin Assignment

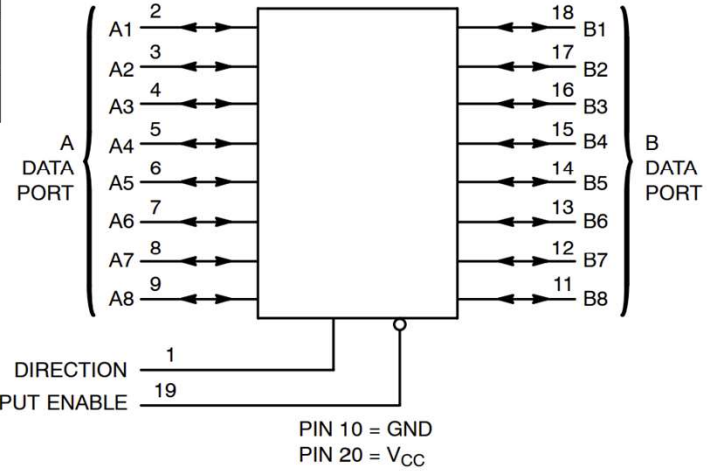


Figure 2. Logic Diagram

273 - Octal D-Type Flip-Flop with Reset

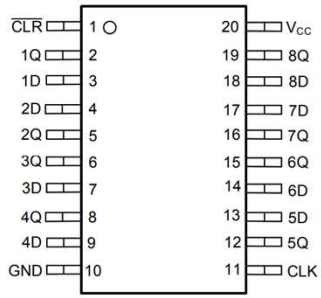
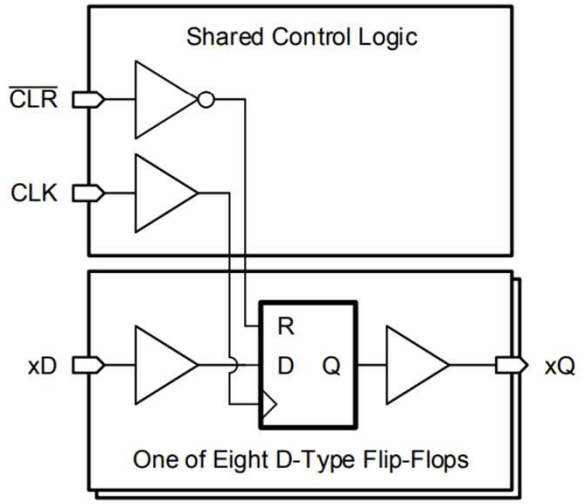


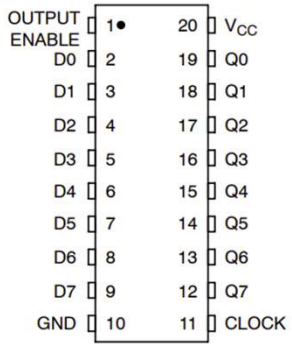
Table 6-1. Truth Table(1)

INPUTS			OUTPUT
RESET ($\overline{CLR}$)	CLOCK CLK	DATA D_n	Q
L	X	X	L
H	↑	H	H
H	↑	L	L
H	L	X	Q_0

(1) H = high voltage level, L = low voltage level, X = don't care, ↑ = transition from low to high level, Q0 = level before the indicated steady-state input conditions were established



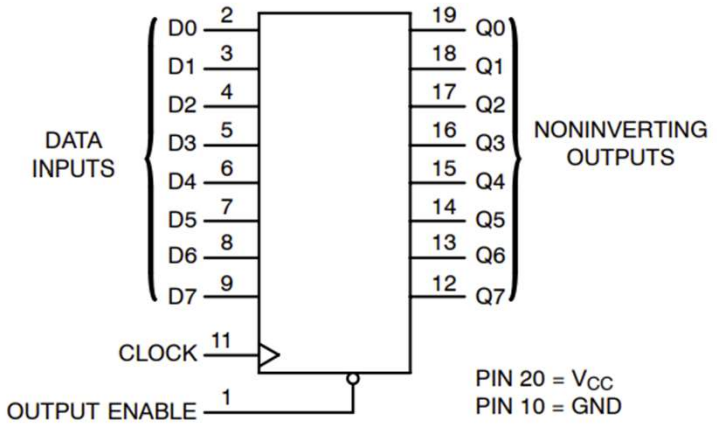
547 - Octal 3–State Noninverting D Flip–Flop



FUNCTION TABLE

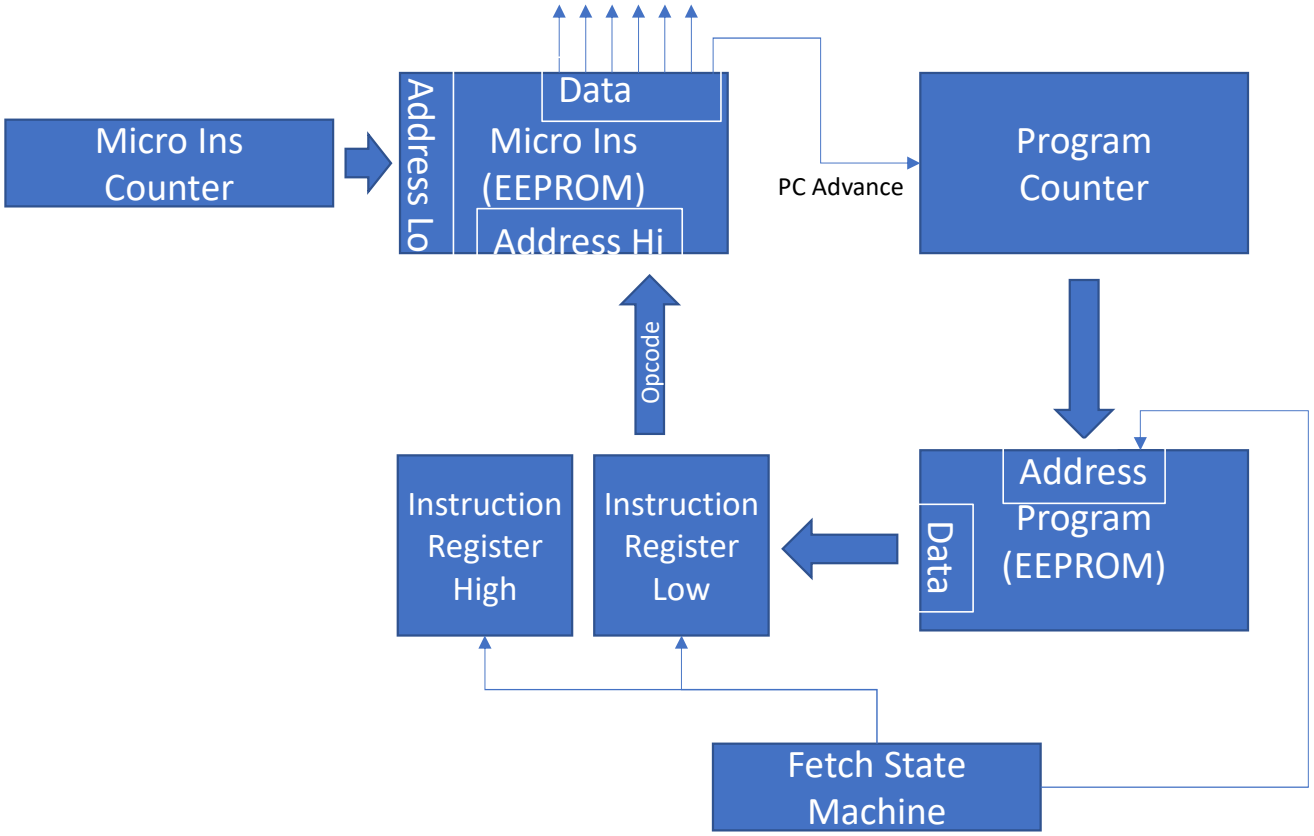
Inputs			Output
OE	**Clock**	**D**	**Q**
L	↑	H	H
L	↑	L	L
L	L,H,↓	X	No Change
H	X	X	Z

X = Don't Care
Z = High Impedance

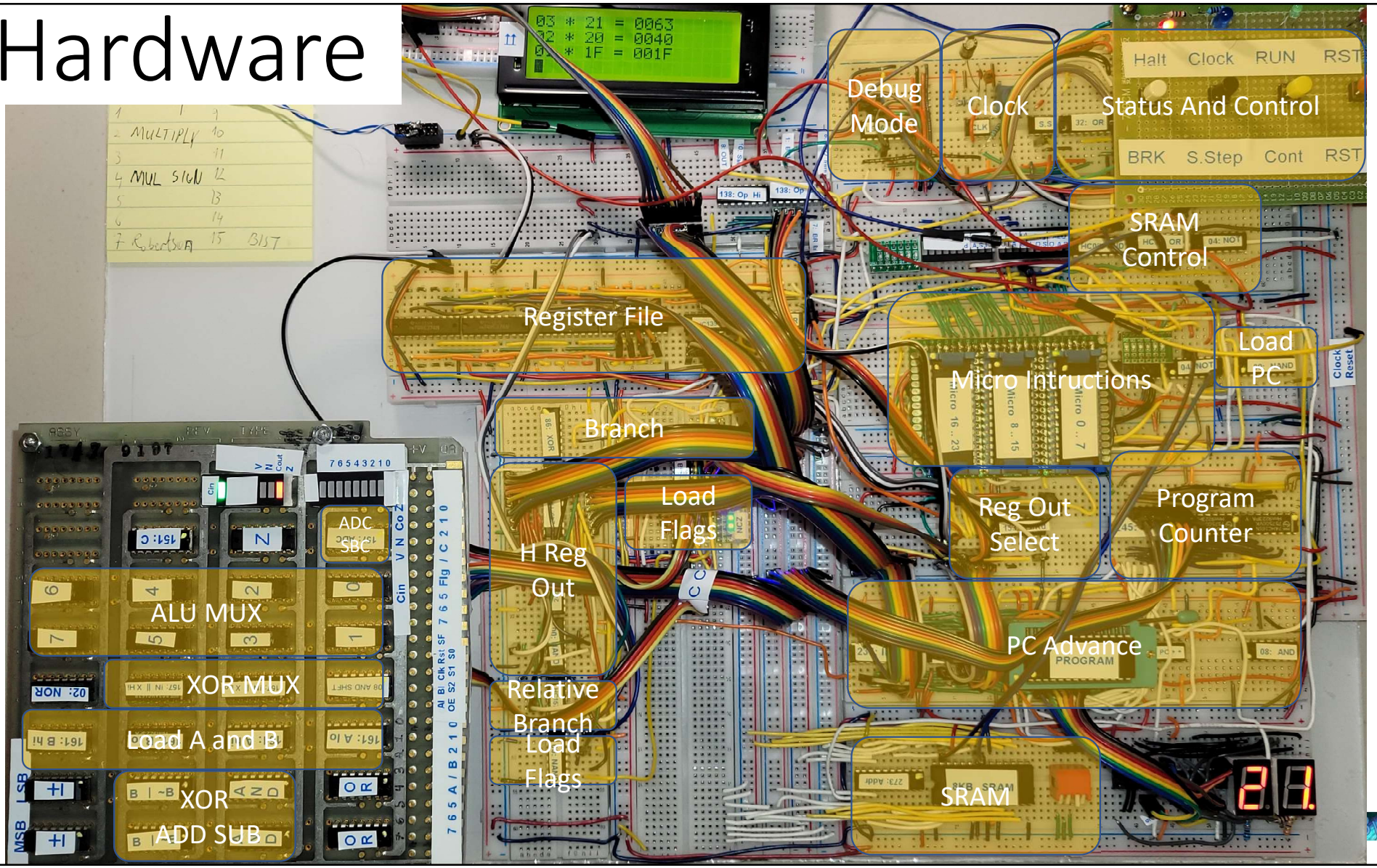


ByteFrost V1.0 – First Working Version

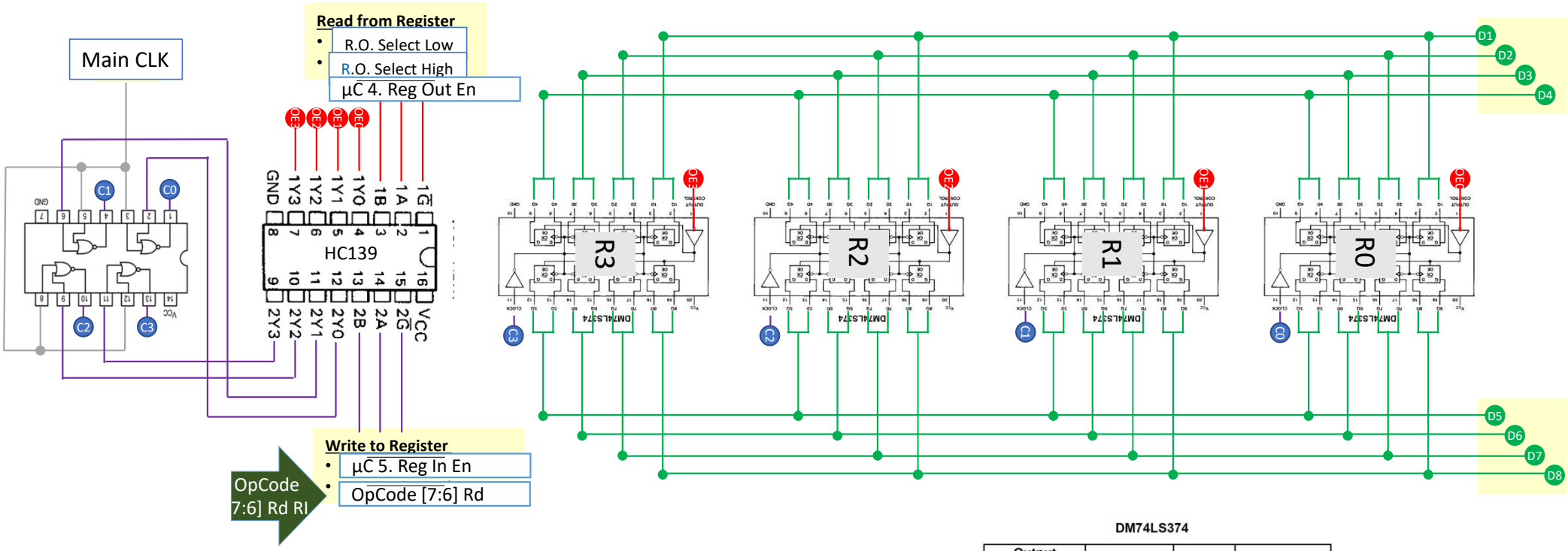
Fetch State Machine Example



Hardware


Debug Mode
Clock
Status And Control
Halt Clock RUN RST
BRK S.Step Cont RST
SRAM Control
Register File
Micro Intructions
Load PC
Branch
Load Flags
H Reg Out
Reg Out Select
Program Counter
ALU MUX
XOR MUX
Load A and B
XOR
ADD SUB
Relative Branch
Load Flags
PC Advance
SRAM

Register File



INPUTS			OUTPUTS			
ENABLE	SELECT					
$\overline{G}$	B	A	Y0	Y1	Y2	Y3
H	X	X	H	H	H	H
L	L	L	L	H	H	H
L	L	H	H	L	H	H
L	H	L	H	H	L	H
L	H	H	H	H	H	L

DM74LS374

Output Control	Clock	D	Output
L	↑	H	H
L	↑	L	L
L	L	X	Q_0
H	X	X	Z

Registers Schematics

Read from Register

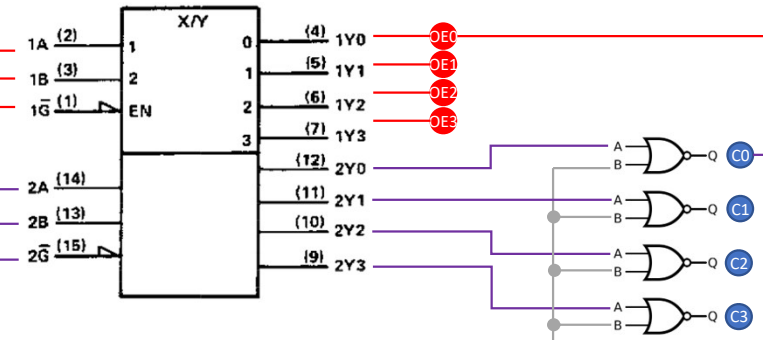
- R.O. Select Low
- R.O. Select High

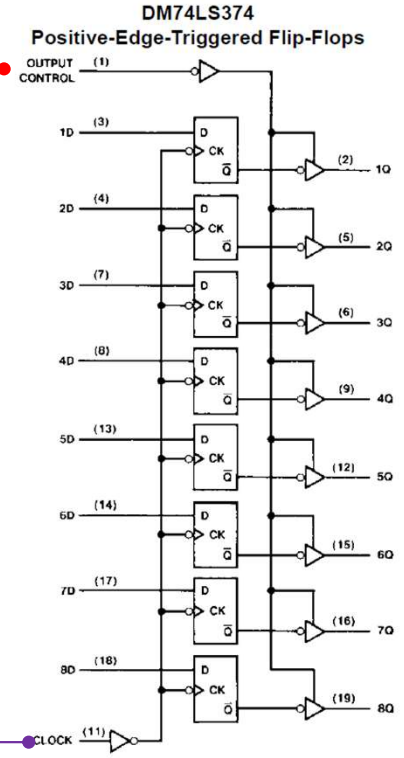
µC 4. Reg Out En

Write to Register

- OpCode [7:6] FGK0
- µC 5. Reg In En

Main CLK





INPUTS			OUTPUTS			
ENABLE	SELECT					
$\overline{G}$	B	A	Y0	Y1	Y2	Y3
H	X	X	H	H	H	H
L	L	L	L	H	H	H
L	L	H	H	L	H	H
L	H	L	H	H	L	H
L	H	H	H	H	H	L

DM74LS374

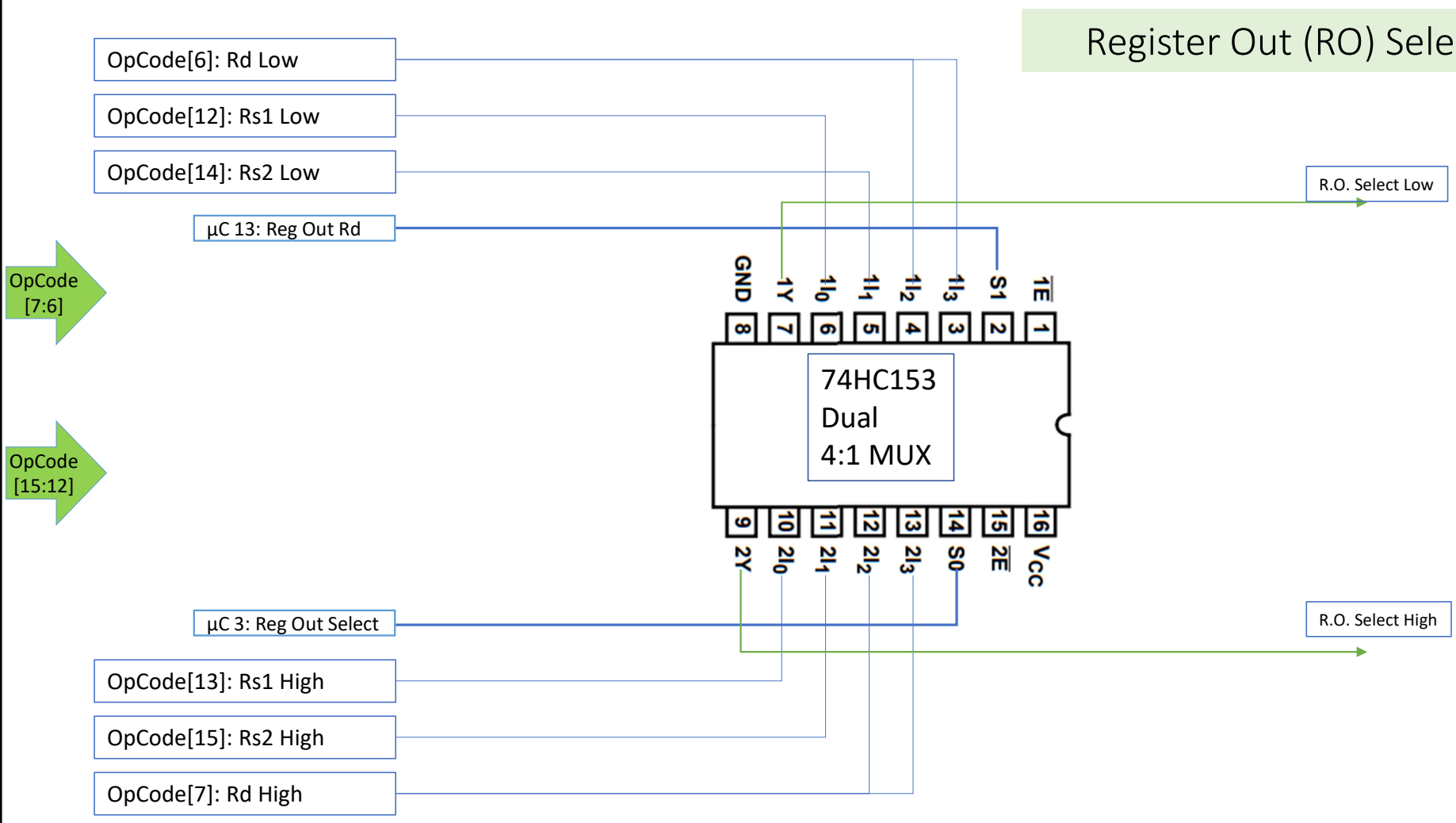
Output Control	Clock	D	Output
L	↑	H	H
L	↑	L	L
L	L	X	Q_0
H	X	X	Z

Slide 55

GK0 Check that bits 6 and 7 of the OpCode (low instruction register) are connected to 2A and 2B

Gil Keidar, 2024-03-10T06:57:17.405

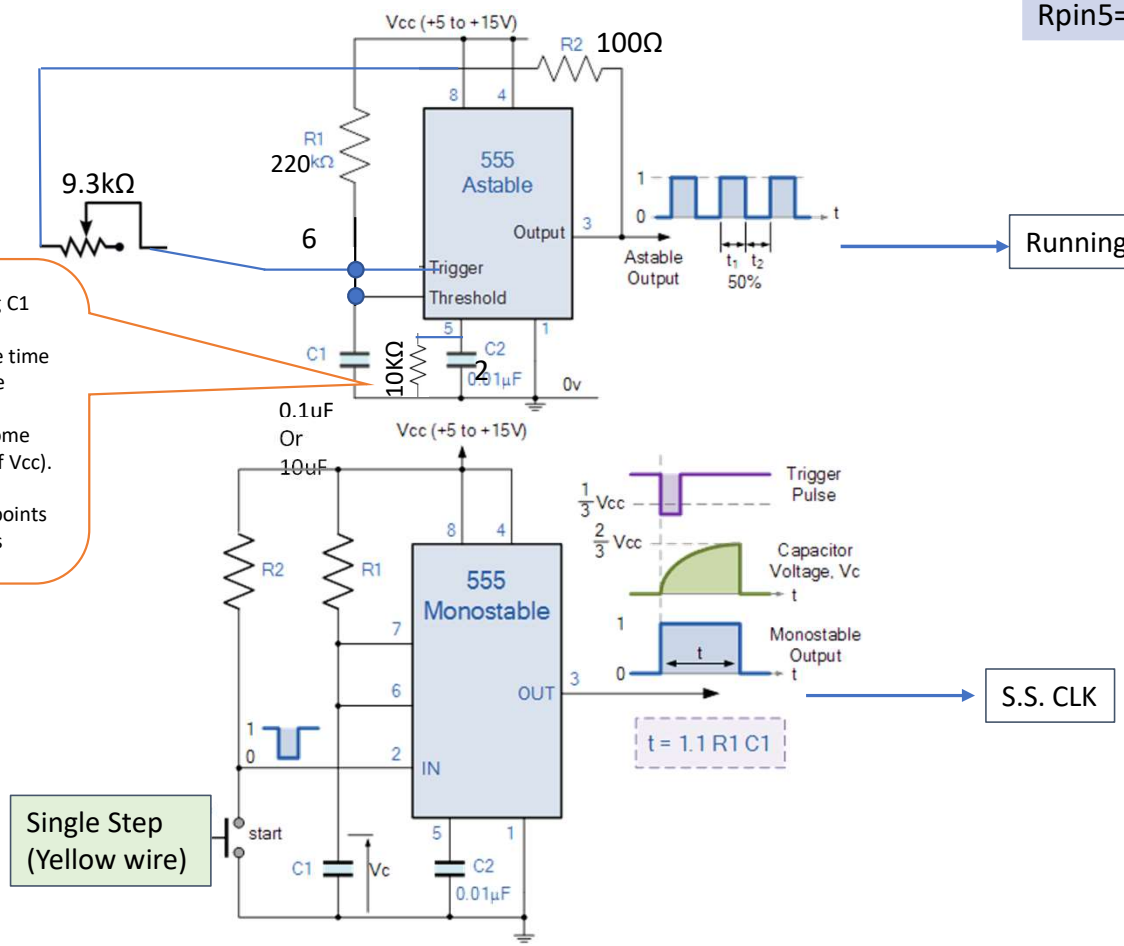
Register Out (RO) Select



Clock

Parts	Min	Frequency
0.01uF (103)	5KHz	420KHz
R2=100		
Rpin5=10K		

GK0

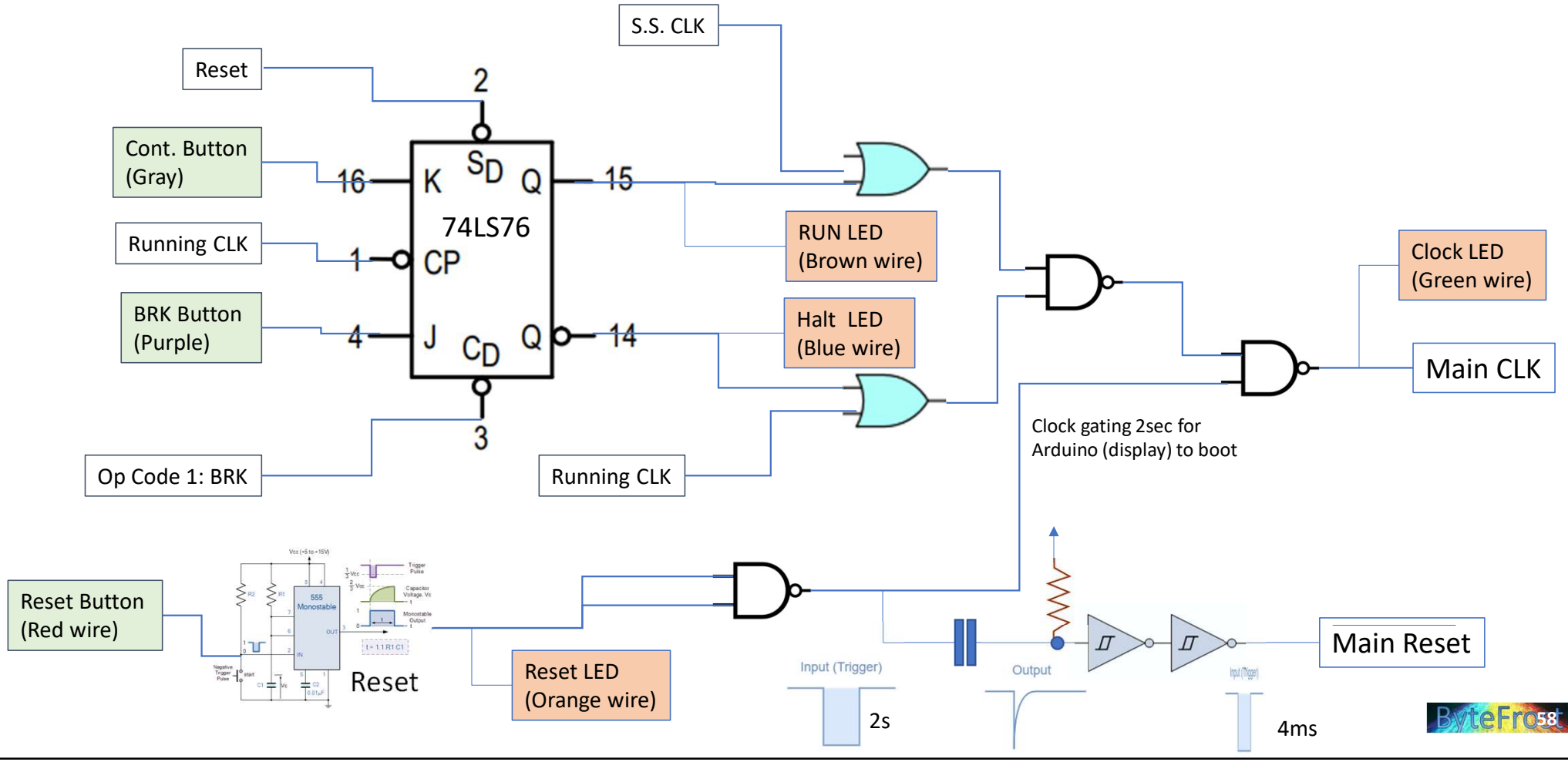


R on pin 5:
The design of using Vout as power for charging C1 generates 50% duty cycle only if Vout==Vcc. In reality, it is lower and the capacitor takes more time to reach the higher trigger point and duty cycle without it is only 33%.
With 10K on pin 5, both triggering points become lower, closer to 1/3 and 2/3 of Vout (instead of Vcc). That provides 50% duty cicle,also increase the frequency because the distance between the points is smaller (increasing the max frequency of this design from 250hHz to 420kHz)

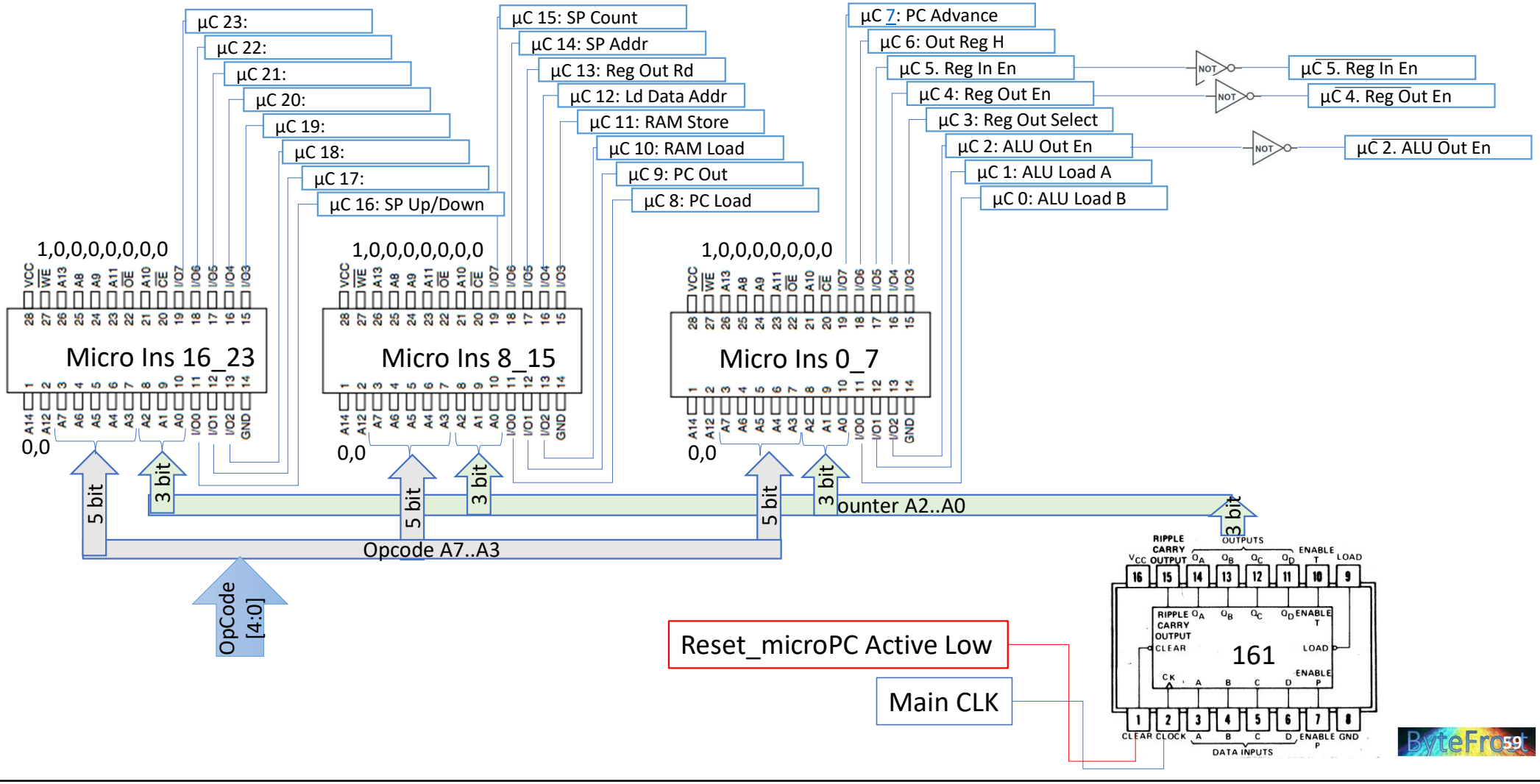
GK0 Verify correct maximum clock frequency using oscilloscope

Gil Keidar, 2024-03-10T06:58:13.803

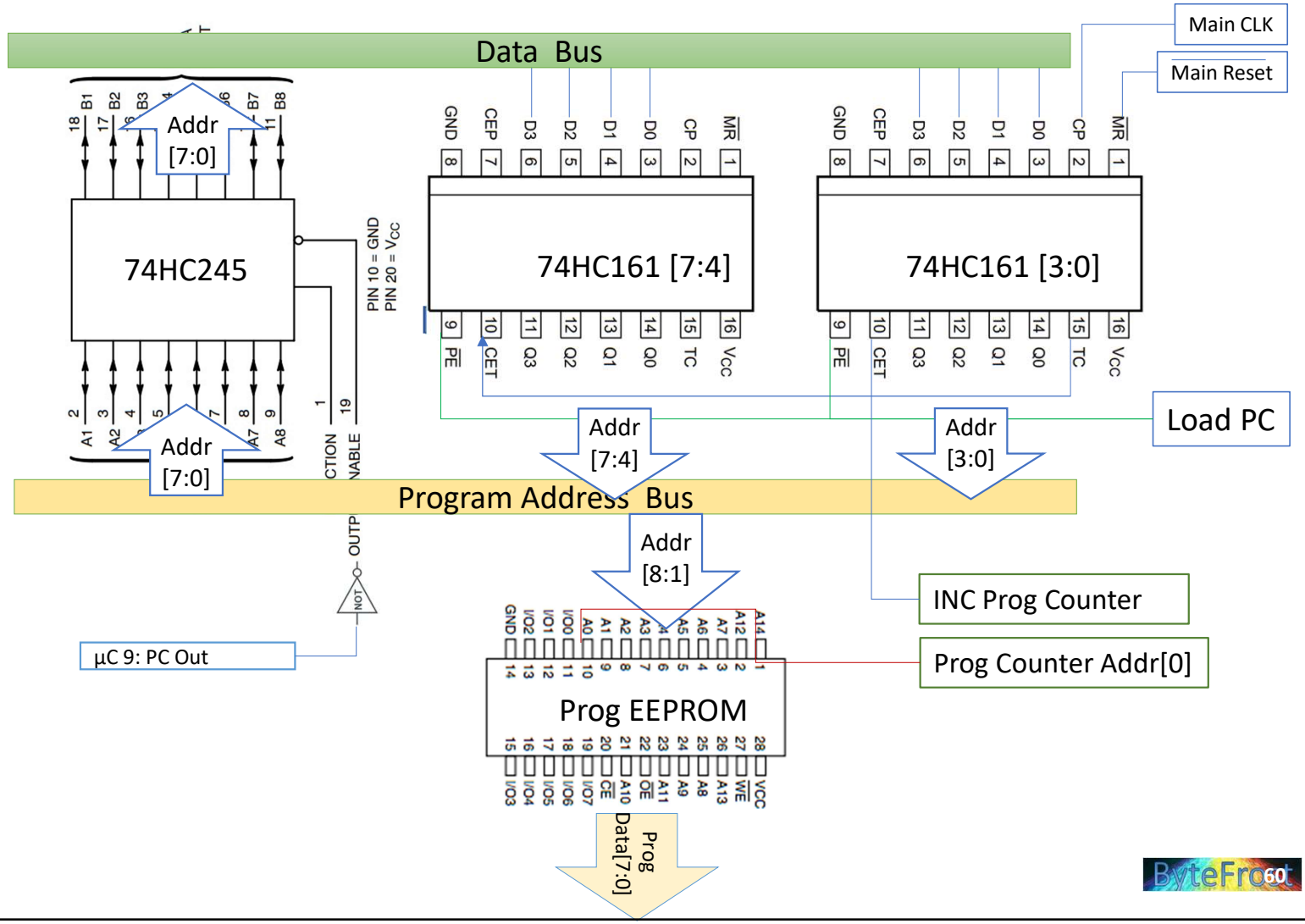
Clock Gating



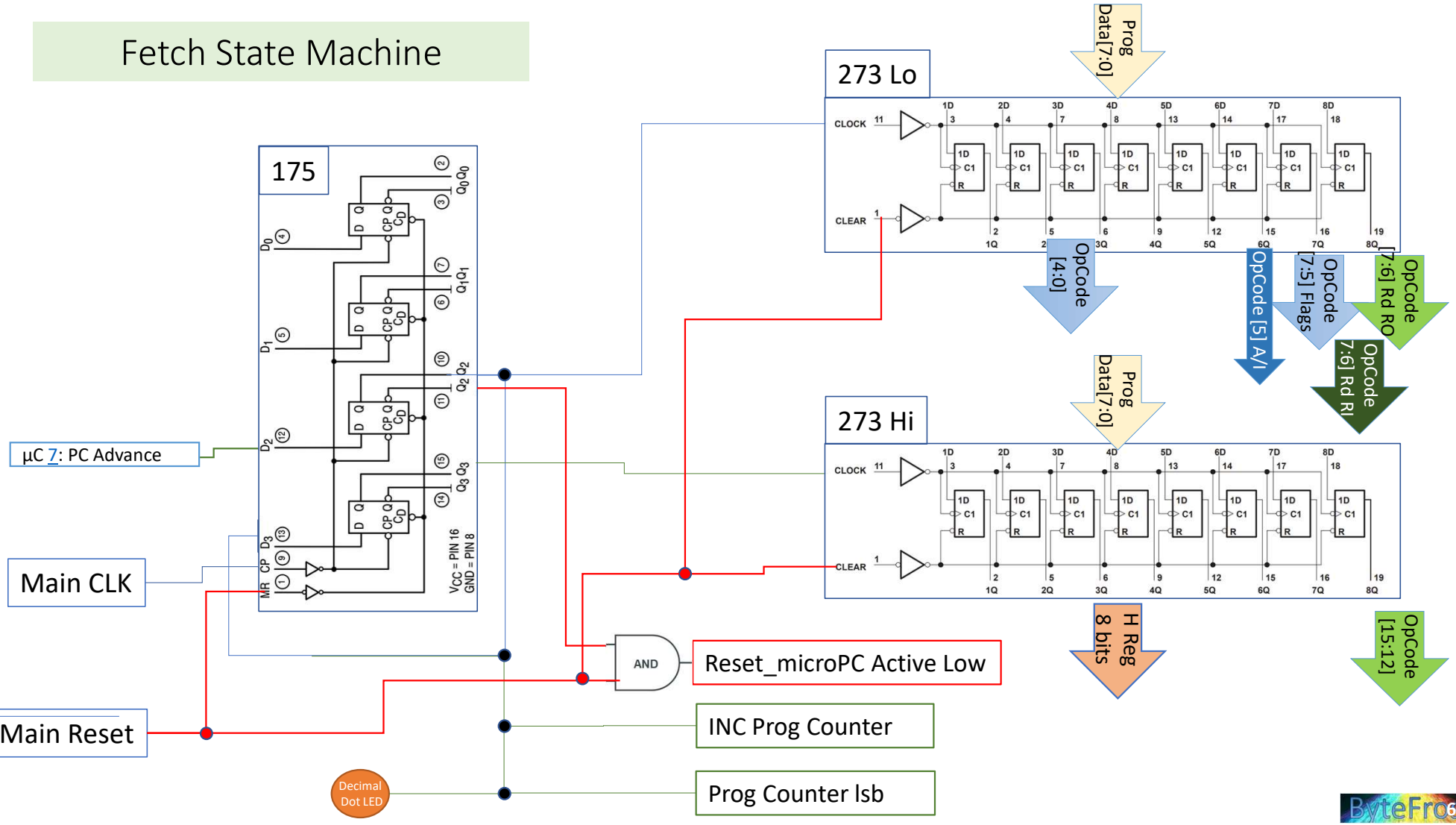
Micro Instructions EEPROMs



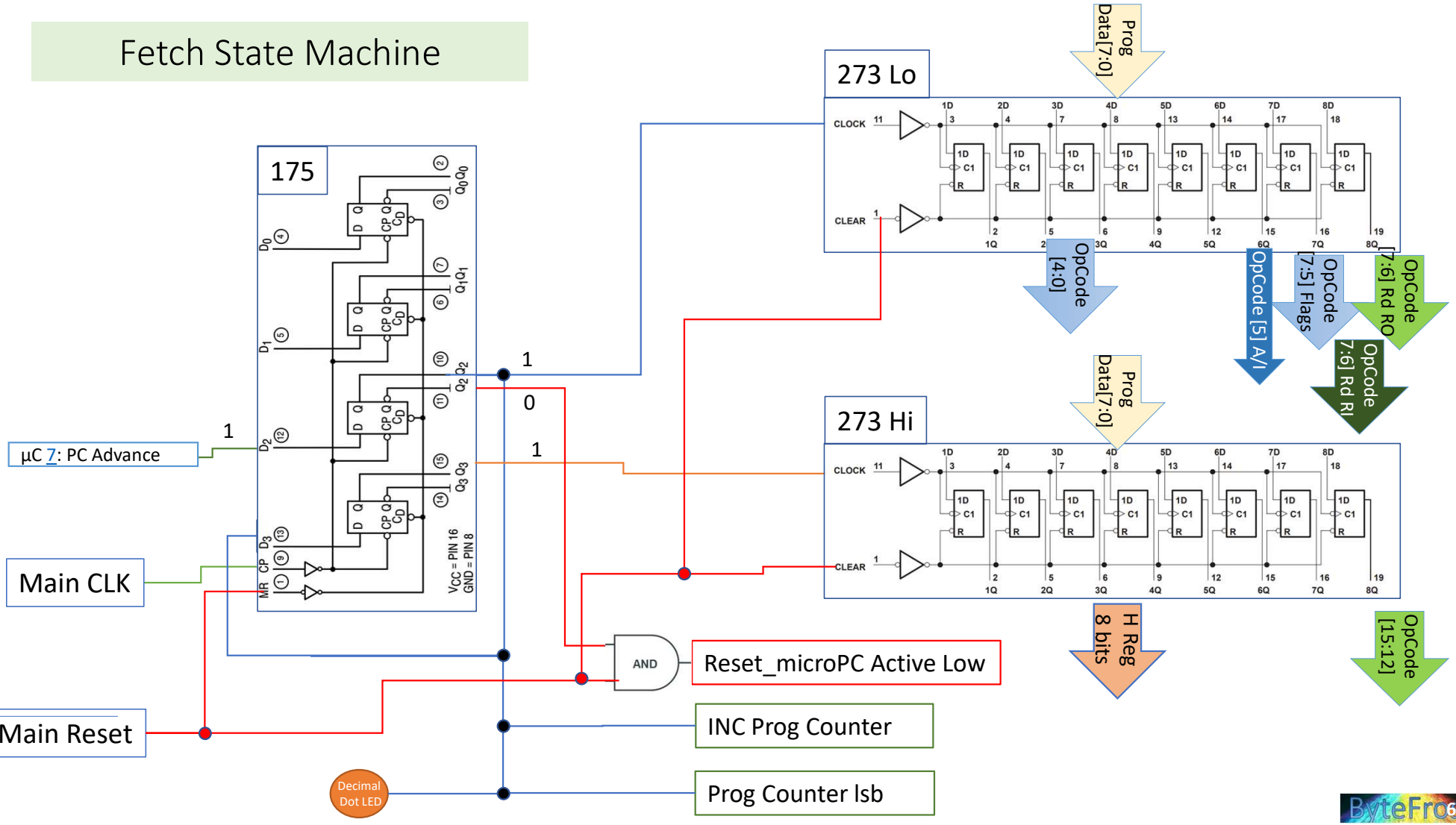
Program Counter



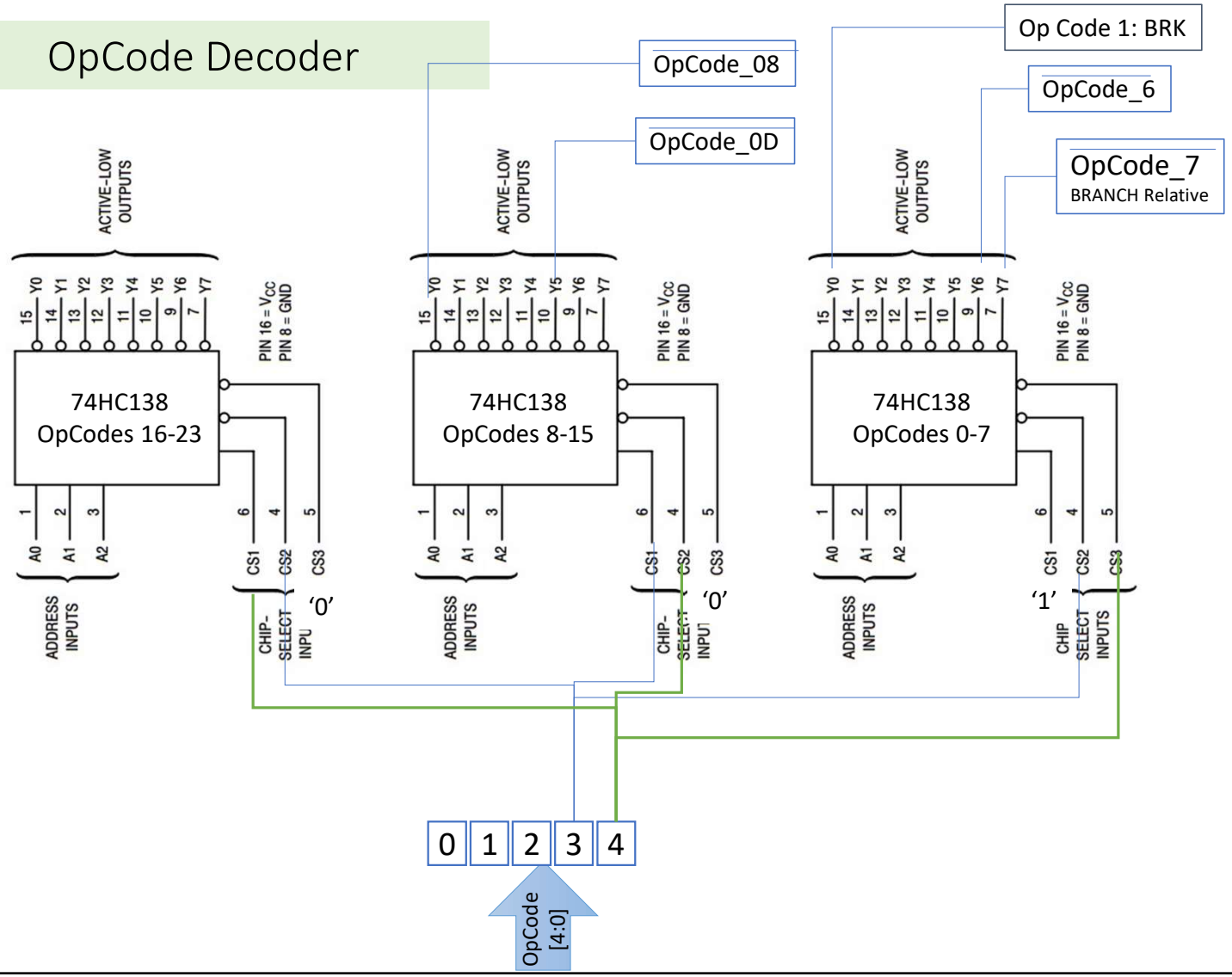
Fetch State Machine



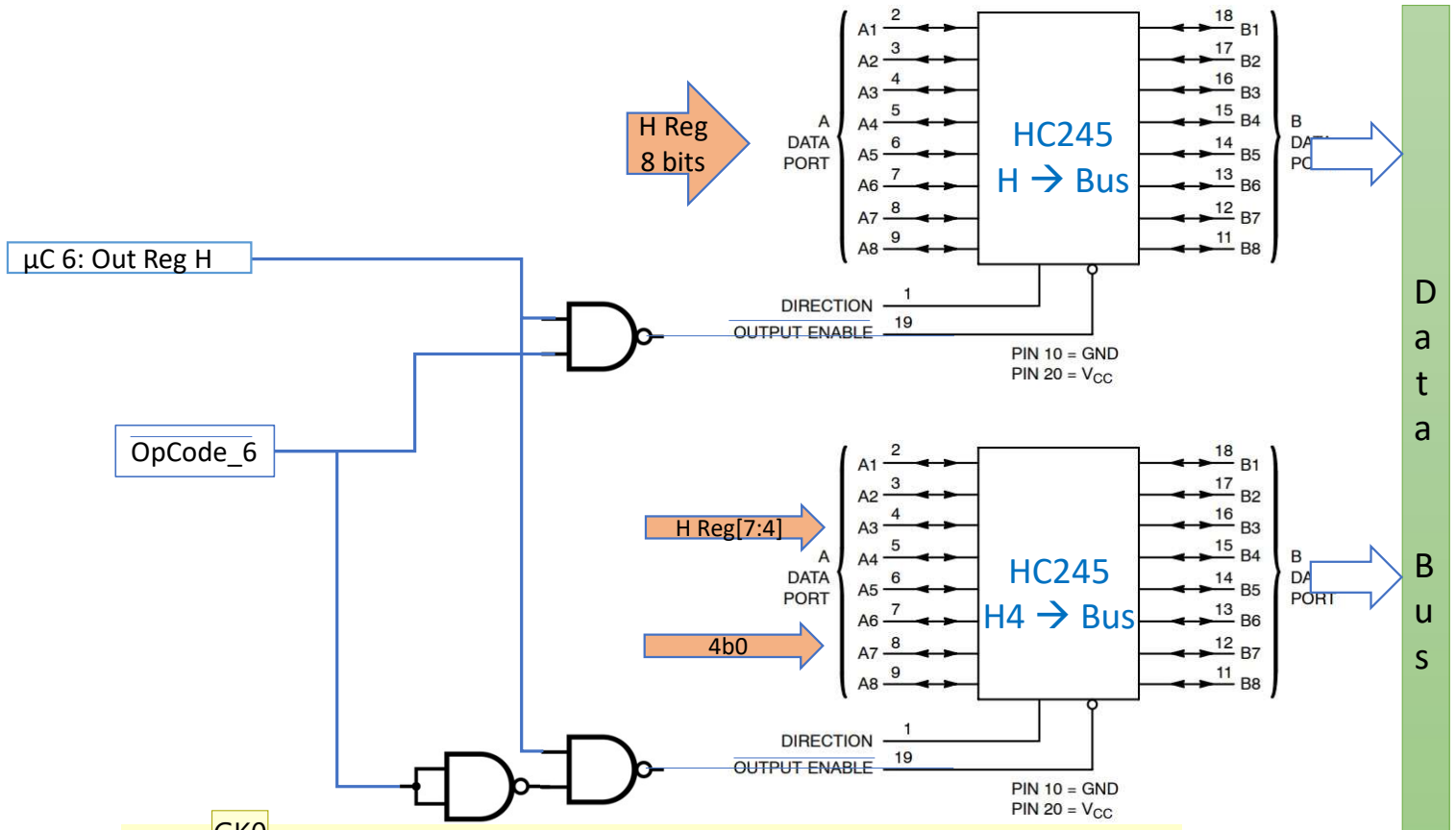
Fetch State Machine



OpCode Decoder



Output H Register to Data Bus



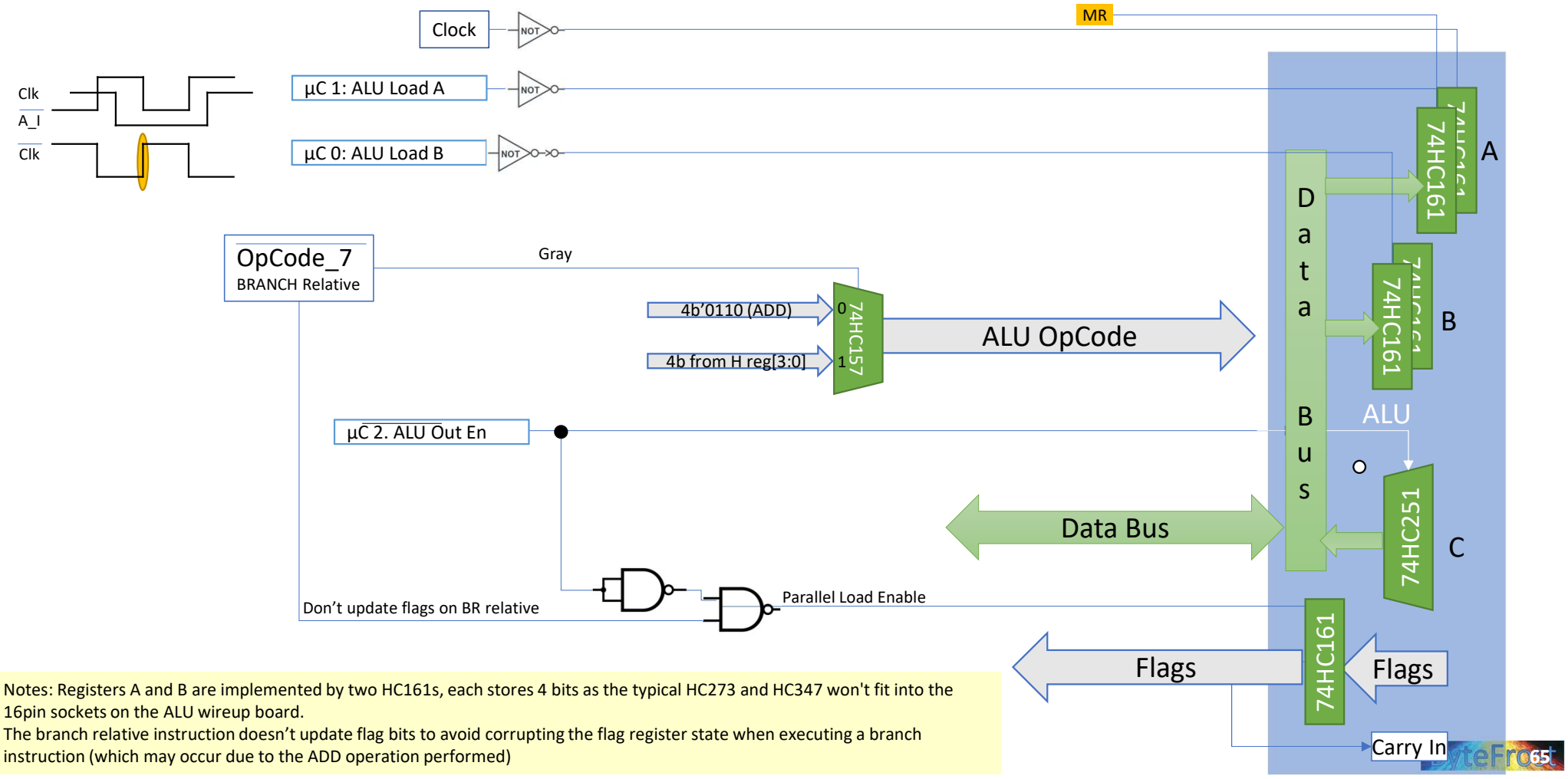
GK0

If opcode is 00110 then zero-extended 4-bit immediate from high instruction register is written to the bus; otherwise, the full 8-bit high instruction register is written to bus as an 8-bit immediate

Slide 64

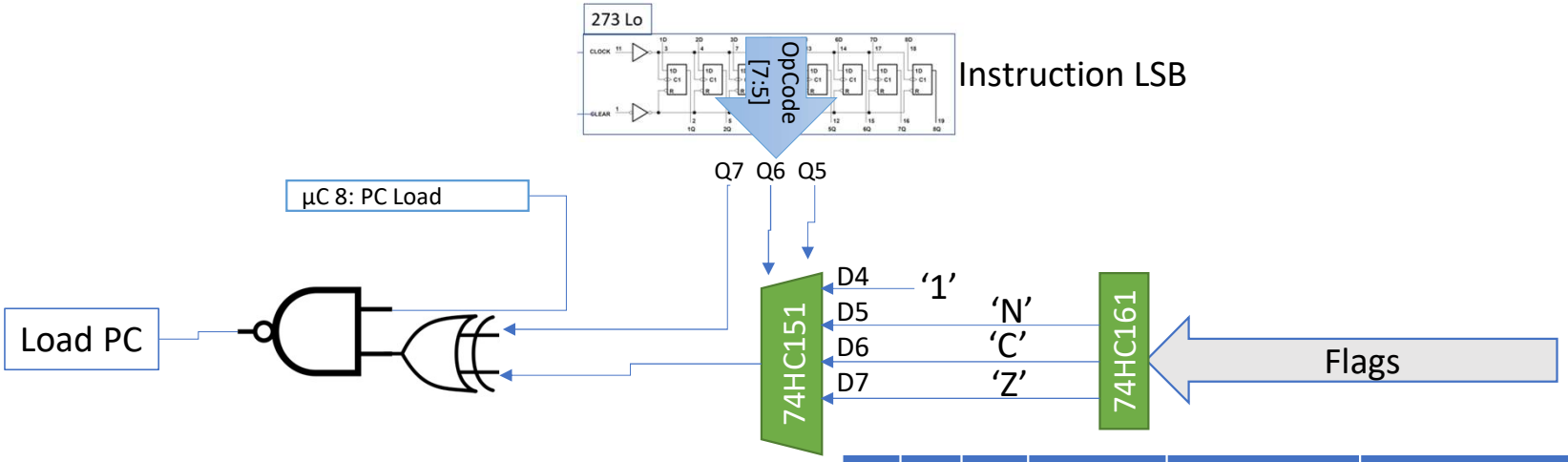
GK0 Note: this might've changed to also include TST immediate(?)
Gil Keidar, 2024-03-10T07:03:07.327

ALU Interface – Flags, A,B Registers, Op Code selection



Notes: Registers A and B are implemented by two HC161s, each stores 4 bits as the typical HC273 and HC347 won't fit into the 16pin sockets on the ALU wireup board.
The branch relative instruction doesn't update flag bits to avoid corrupting the flag register state when executing a branch instruction (which may occur due to the ADD operation performed)

Branch

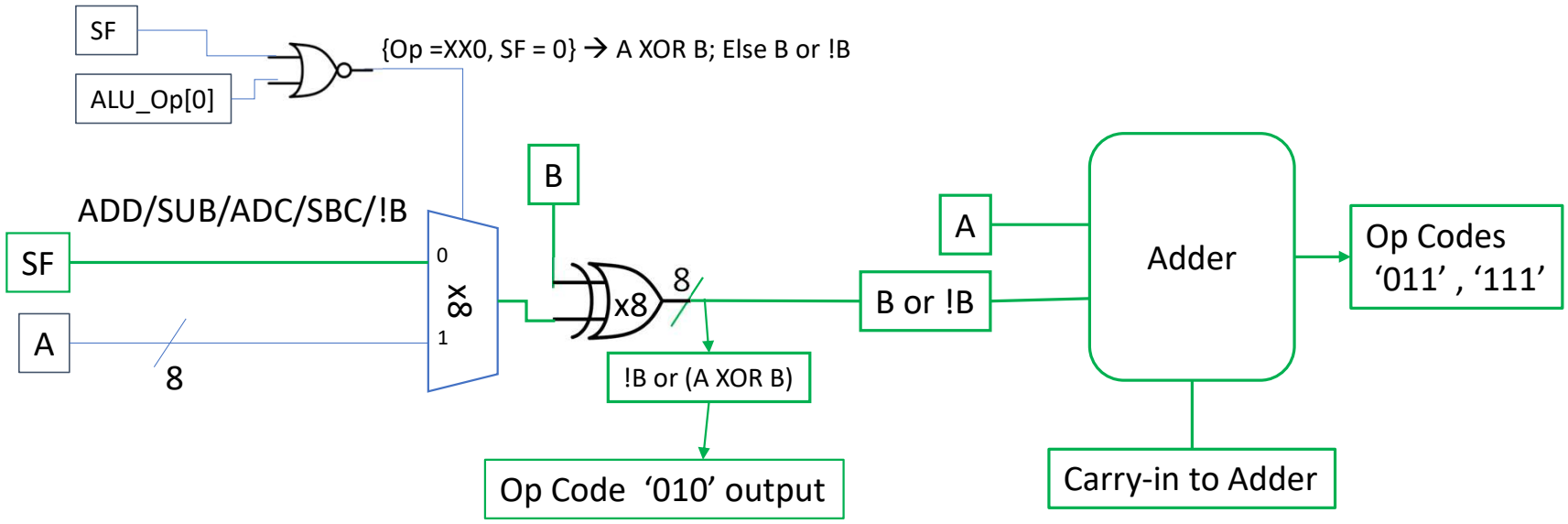


C2	C1	C0	Condition	Instruction LSB	Equivalent Assembly Instruction
0	0	0	JMP	05	JMP
0	0	1	N	25	BMI – Branch on Minus
0	1	0	C	45	BCS – Branch Carry Set
0	1	1	Z	65	BEQ – Branch Equal
1	0	0	Never	85	NOP
1	0	1	!N	A5	BPL – Branch on Plus
1	1	0	!C	C5	BCC – Branch Carry Clear
1	1	1	!Z	E5	BNE

ALU OpCodes 2,3,7

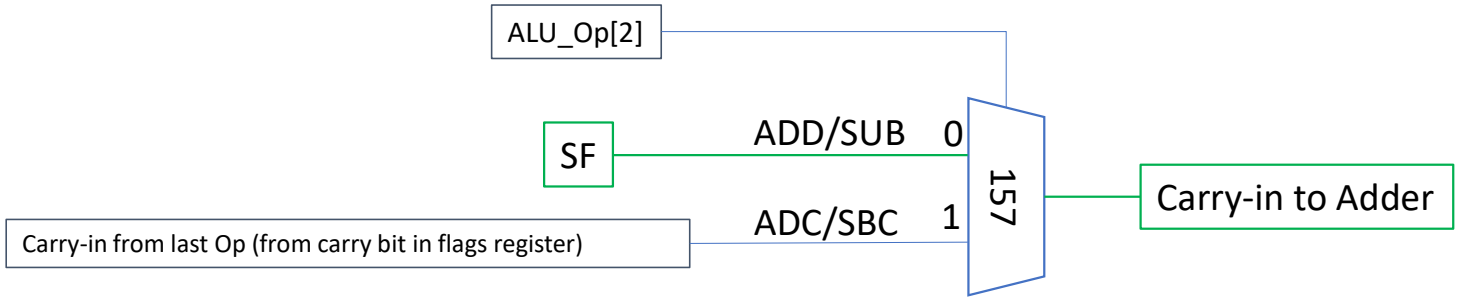
Op Select	Special Func. = 0		Special Func. = 1		Flag in
010 - 2	A XOR B	XOR: 4	!B	NOT: 5	-
011 - 3	A + B	ADD: 6	A – B	SUB: 7	-
111- 7	A + B + Cin	ADC: E	A – B + Cin	SBC: F	Cin

- ALU opcode(010), SF(0 / 1) – A XOR B / !B
- No change to the source SF for ADD/SUB/ADC/SBC/!B

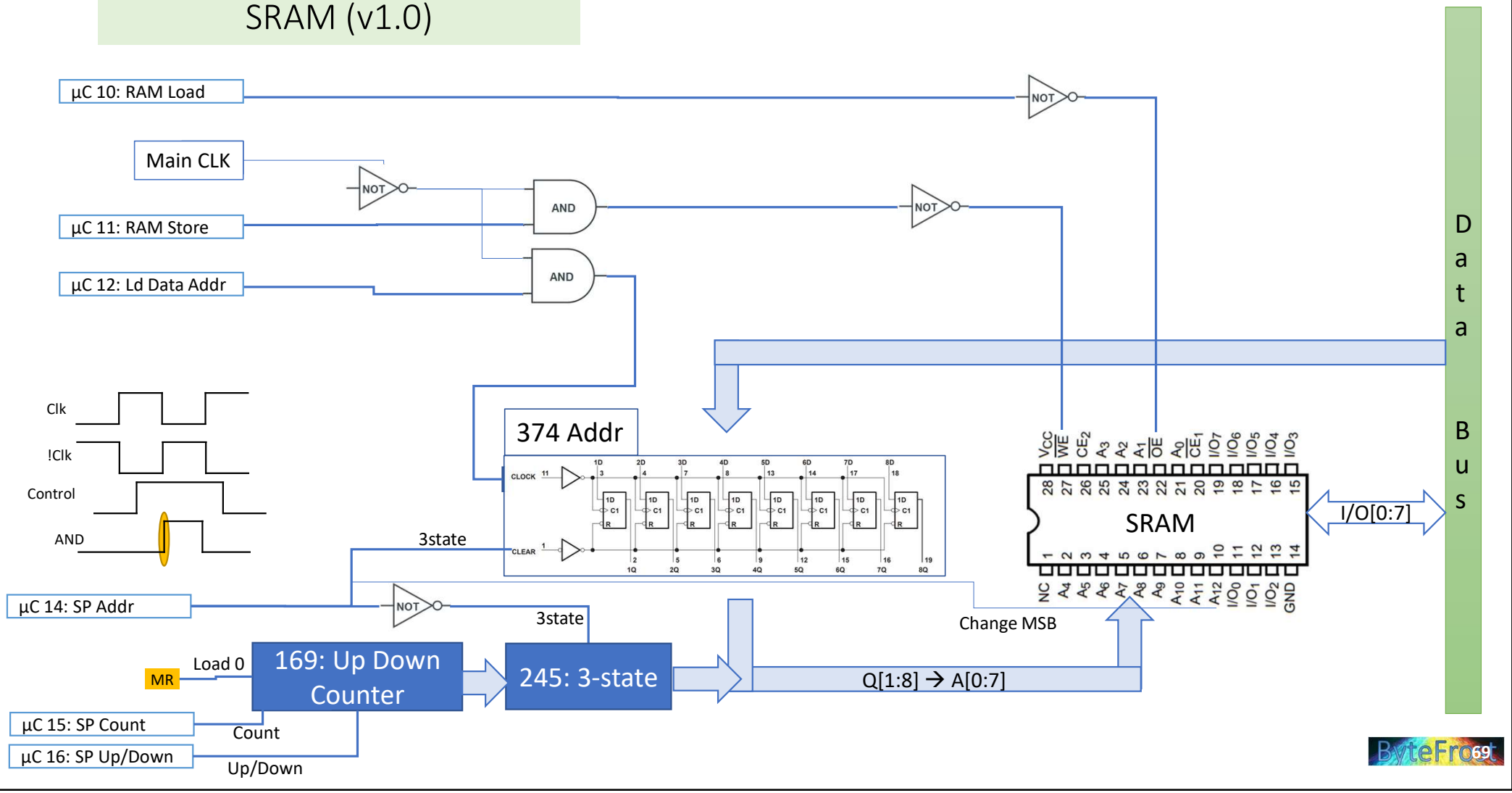


ALU: ADD Carry/SUB Carry

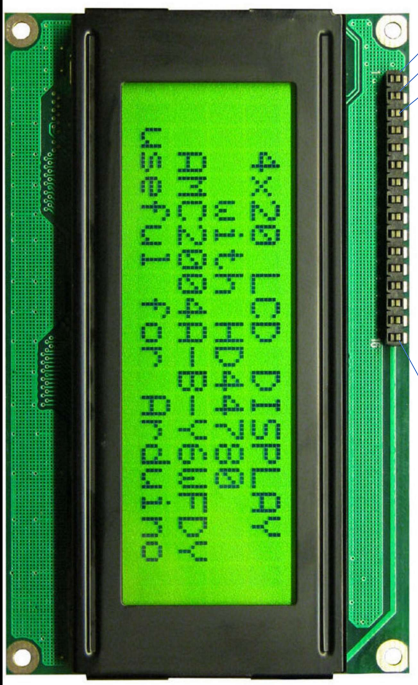
- Note the difference between the ALU function opcodes for ADD / SUB and ADC / SBC:
  - ALU opcode(011), SF(0 / 1) – ADD/SUB
  - ALU opcode(111), SF(0 / 1) – ADC/SBC
- Note: The 6502 ISA only supports ADC/SBC (along with Set Carry and Reset Carry)



SRAM (v1.0)


µC 10: RAM Load
Main CLK
µC 11: RAM Store
µC 12: Ld Data Addr
NOT
AND
AND
NOT
NOT
Clk
!Clk
Control
AND
374 Addr
CLOCK
CLEAR
3state
µC 14: SP Addr
NOT
3state
Change MSB
MR
Load 0
169: Up Down Counter
245: 3-state
Q[1:8] → A[0:7]
µC 15: SP Count
Count
µC 16: SP Up/Down
Up/Down
SRAM
I/O[0:7]
Data Bus
ByteFrost

Display



VSS = Connect to Ground
VDD = Connect to 5V
VO = Contrast Adjustment (Connect to pot)
RS = Reg Select. H=Data, L= Instruction
RW = Read / Write. Ground for write
E = Enable. Data latched on falling edge
D0 = Data Bit 0
D1 = Data Bit 1
D2 = Data Bit 2
D3 = Data Bit 3
D4 = Data Bit 4
D5 = Data Bit 5
D6 = Data Bit 6
D7 = Data Bit 7
A = Backlight Anode. Connect to 5V
K = Backlight Cathode. Connect to ground

D12
D11
D0
D1
D13
D3
D4
D5
D6
D7

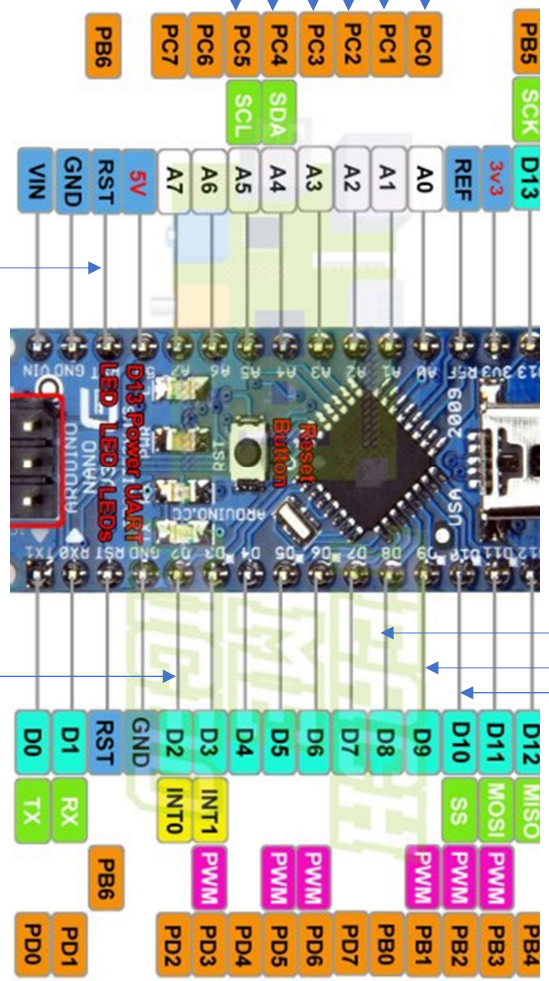
MR

Disp Char 5 4 3 2 1 0 6 7

INT0

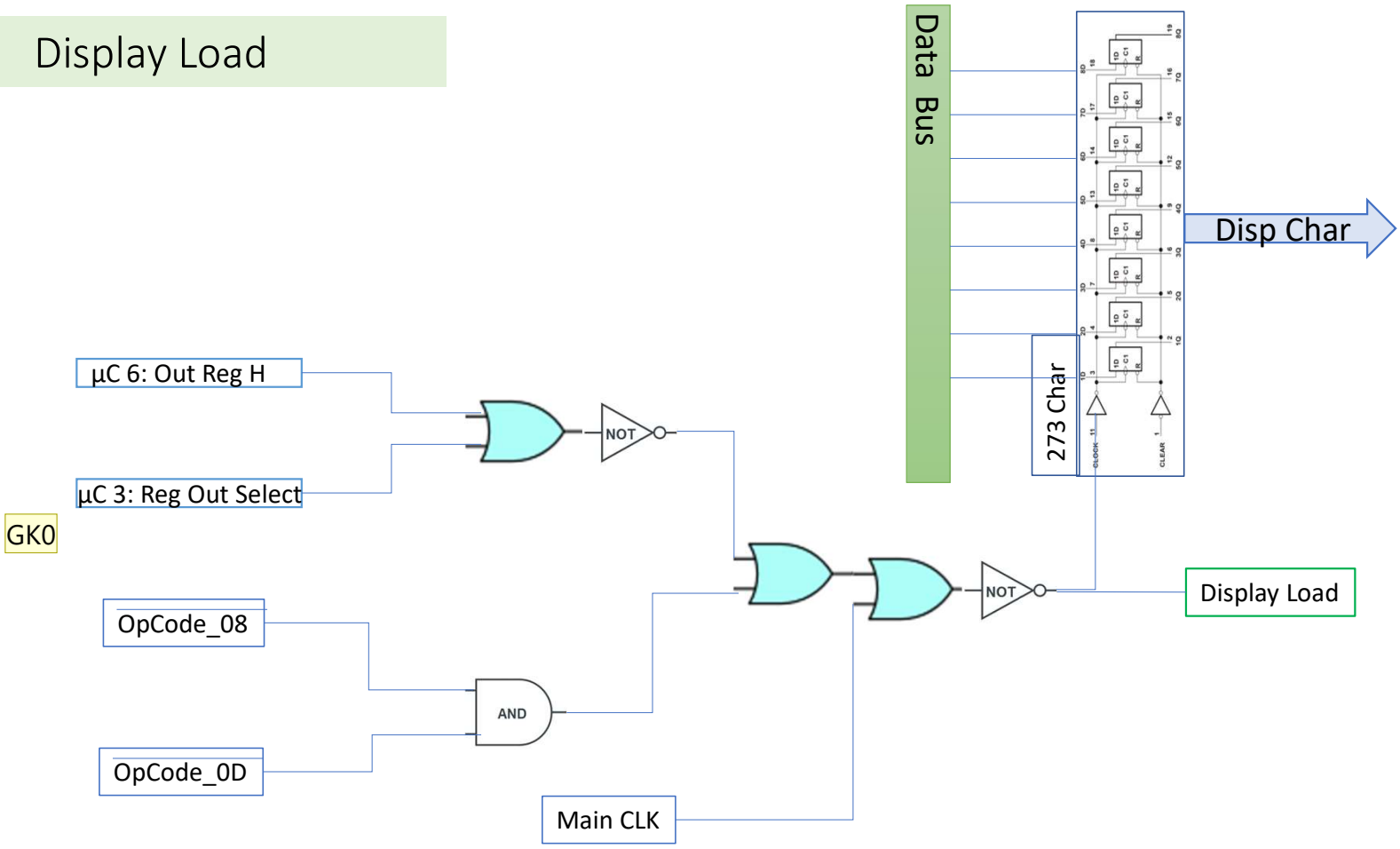
Display Load

OpCode [5] A/I



ByteFrost

Display Load



ByteFrost

Slide 71

GK0 Check whether the input here is uC3 and not uC4

Gil Keidar, 2024-03-10T07:44:26.876

Op Code
&
Micro Instruction

Assembler Syntax

An assembly file is composed of lines with the following form:

[INSTR] [PARAM 1] [, PARAM 2] [, PARAM 3] // Comment

Misc. Instruction	Meaning	Format
NOP	No operation	NOP
BRK	Stop	BRK
OUT	Print 8-bit value to output	OUT Rs, #imm

Register Instruction	Meaning	Format
LDR	Load register with 8-bit immediate	LDR Rd, #imm
MOV	Copy register value to another register	MOV Rd, Rs

ALU Instruction	Meaning	Format
OR	Bitwise OR	OR Rd, Rs1, Rs2
AND	Bitwise AND	AND Rd, Rs1, Rs2
XOR	Bitwise XOR	XOR Rd, Rs1, Rs2
NOT	Bitwise NOT	NOT Rd, Rs1
ADD	Addition	ADD Rd, Rs1, Rs2
SUB	2's Complement subtraction	SUB Rd1, Rs1, Rs2 (Rd1 = Rs1 – Rs2)
LSL	Bitwise logical shift left by 1 bit	LSL Rd, Rs1
ASR	Arithmetic shift right by 1 bit	ASR Rd, Rs1
ADC	Add with carry	ADC Rd, Rs1, Rs2
SUC	Subtract with borrow (using carry flag)	SUC Rd, Rs1, Rs2 (Rd1 = Rs1 – Rs2)

Branch Instruction	Meaning	Format
JMP	Unconditional branch	A + I: JMP #imm A + R: R + I: R + R:
BIN	Branch If Negative	A + I: BIN #imm
BIC	Branch if Carry	A + I: BIC #imm
BIZ	Branch if Zero	A + I: BIZ #imm
BNN	Branch if not Negative	A + I: BNN #imm
BNC	Branch if not Carry	A + I: BNC #imm
BNZ	Branch if not Zero	A + I: BNZ #imm

Instruction set families

0 15

Type	0 1 2 3 4	5	6 7	8 9 10 11	12 13	14 15
ALU	OpCode	X	Rd	Function	Rs1	Rs2

Type	0 1 2 3 4	5	6 7	8 9 10 11	12 13	14 15
Load Imm	OpCode	X	Rd	Immediate (8-bit)		

Type	0 1 2 3 4	5	6 7	8 9 10 11	12 13	14 15
Load Reg	OpCode	X	Rd	XXXX	Rs1	XX

Instruction set

Op code	Instruction	Details
0000	NOP	No operation
0001	BRK	Break
0010	ALU (Format ALU Rd, Rs1, Rs2)	ALU instruction
0011	LDR	Load Register (immediate, ~~other register~~) ?
0100	MOV	Copy Register (MOV instruction in ARM)
0101	Branch Absolute	Jump to an address (8bit)
0110	ALU Immediate (Format: ALU Rd, Rd, imm)	ALU instruction with value from immediate
0111	BRANCH Relative	Relative Branch (+127 to -128)
1000	OUT reg	Send a register as a character or integer to the display
1001	LMA	Load register from address
1010	SMA	Store register to address
1011	LMR	Load register from address in another register
1100	SMR	Store register into an address in another register
1101	GSB	Jump, store current PC in stack
1110	RTN	Load PC from stack
1111	OUT Immediate	Send Reg H as a character or integer to display

Phase 0	**NOP, BRK**
Phase 1	Fib, Mult
Phase 2	Self Test
Phase 3	Display, 8 Queen
Completed	

Micro-Instructions Control: NOP Instruction (00000)

0

Bit	Operation
16	Stack Pointer Increment / Decrement (0: decrease / 1: increase)
15	Stack Pointer Count
14	RAM Address Select (0: Address register / 1: Stack pointer)
13	Use Rd as Source
12	Lower Address Register Load
11	Mem Write
10	Mem Read
9	PC Out
8	PC Load
7	PC advance
6	Program Register H Write to Bus (parameter)
5	Register File Input Enable
4	Register File Output Enable
3	Register File Output Select (parameter – 0 -> Rs1, 1 -> Rs2)
2	ALU output enable
1	ALU load register A
0	ALU load register B

Cycle #	Instruction	Explanation
0	00000000 = 0x00	
1	10000000 = 0x80	Increment PC
2		
3		
4		
5		
6		
7		

GK0

NOP

Slide 76

GK0 Can this be optimized to reduce one clock cycle? I.e., on cycle 0, have the microinstruction 0x80?
Gil Keidar, 2024-03-10T07:17:46.231

Micro-Instructions Control: BREAK (00001)

1

Bit	Operation
16	Stack Pointer Increment / Decrement (0: decrease / 1: increase)
15	Stack Pointer Count
14	RAM Address Select (0: Address register / 1: Stack pointer)
13	Use Rd as Source
12	Lower Address Register Load
11	Mem Write
10	Mem Read
9	PC Out
8	PC Load
7	PC advance
6	Program Register H Write to Bus (parameter)
5	Register File Input Enable
4	Register File Output Enable
3	Register File Output Select (parameter – 0 -> Rs1, 1 -> Rs2)
2	ALU output enable
1	ALU load register A
0	ALU load register B

Cycle #	Instruction	Explanation
0	00000000 = 0x00	Do nothing
1	10000000 = 0x80	Increment PC
2		
3		
4		
5		
6		
7		

Note: BRK has the same microcode as NOP, but the opcode decoder halts the clock when the opcode 00001 is detected. Stepping through it manually (via Single-Step / Run button press) resumes the clock. Hence, BRK acts as a hardware breakpoint.

BRK

ALU with changes

Op Select	Special Func. = 0		Special Func. = 1		Flag in	Flags Out
0	A OR B	OR: 0	A OR B		-	Z, N
1	A AND B	AND: 2	A AND B		-	
2	A XOR B	XOR: 4	~B	NOT: 5	-	Z, N
3	A + B	ADD: 6	A – B	SUB: 7	-	Z, N, Co
4	Shift Left A A[0]=0	ASL: 8	Rotate left A A[0]=Cin	ROL: 9	Cin (Rotate)	Z, N=A[6], Co=A[7]
5	Shift Right A	LSR: A	Arithmetic S.R. A	ASR: B	-	Z, N, Co=A[0]
6	Rotate Right	ROR: C	Cin -> A-> Cout		Cin	Z, N, Co=A[0]
7	A + B + Cin	ADC: E	A – B + Cin	SBC: F	Cin	Z, N, Co

Proposal	# of IC slots	Description	Op Code
OR and ADD share Op Code (low priority)	2x 74LS157	Special Function select OR or AND	Release op code
Move A, B Regs (require replacement of 74LS251 with 74HC251)	4x 74LS161 1x 74LS00 NAND	Reduce interface to one data bus instead of A, B, C (NAND for clock gate/invert)	No change

Micro-Instructions Control: ALU (00010)

2

Bit	Operation
16	Stack Pointer Increment / Decrement (0: decrease / 1: increase)
15	Stack Pointer Count
14	RAM Address Select (0: Address register / 1: Stack pointer)
13	Use Rd as Source
12	Lower Address Register Load
11	Mem Write
10	Mem Read
9	PC Out
8	PC Load
7	PC advance
6	Program Register H Write to Bus (parameter)
5	Register File Input Enable
4	Register File Output Enable
3	Register File Output Select (parameter – 0 -> Rs1, 1 -> Rs2)
2	ALU output enable
1	ALU load register A
0	ALU load register B

Cycle #	Instruction	Explanation
0	00010010 = 0x12	Load Rs1 to reg A
1	00011001 = 0x19	Load Rs2 to reg B
2	00100100 = 0x24	Output and Read to Rd
3	10000000 = 0x80	Increment PC
4		
5		
6		
7		

```
ADD Rd, Rs1, Rs2
```

Type	15 14	13 12	11 10 9 8	7 6	5 4	3 2 1 0
ALU	Rs2	Rs1	S2 S1 S0 SF (Function sel)	Rd	XX	OpCode

Micro-Instructions Control: Load Immediate to Reg (00011)

3

Bit	Operation
16	Stack Pointer Increment / Decrement (0: decrease / 1: increase)
15	Stack Pointer Count
14	RAM Address Select (0: Address register / 1: Stack pointer)
13	Use Rd as Source
12	Lower Address Register Load
11	Mem Write
10	Mem Read
9	PC Out
8	PC Load
7	PC advance
6	Program Register H Write to Bus (parameter)
5	Register File Input Enable
4	Register File Output Enable
3	Register File Output Select (parameter – 0 -> Rs1, 1 -> Rs2)
2	ALU output enable
1	ALU load register A
0	ALU load register B

Cycle #	Instruction	Explanation
0	01100000 = 0x60	Write imm to bus and Rd read
1	10000000 = 0x80	Increment PC
2		
3		
4		
5		
6		
7		

```
LDR Rd, #0x00
```

Type	15 14	13 12	11 10 9 8	7 6	5	4 3 2 1 0
Load Imm	Immediate (8-bit)			Rd	X	OpCode

Micro-Instructions Control: MOV (00100)

4

Bit	Operation
16	Stack Pointer Increment / Decrement (0: decrease / 1: increase)
15	Stack Pointer Count
14	RAM Address Select (0: Address register / 1: Stack pointer)
13	Use Rd as Source
12	Lower Address Register Load
11	Mem Write
10	Mem Read
9	PC Out
8	PC Load
7	PC advance
6	Program Register H Write to Bus (parameter)
5	Register File Input Enable
4	Register File Output Enable
3	Register File Output Select (parameter – 0 -> Rs1, 1 -> Rs2)
2	ALU output enable
1	ALU load register A
0	ALU load register B

Cycle #	Instruction	Explanation
0	00110000 = 0x30	Write Rs1 to bus and Rd reads
1	10000000 = 0x80	Increment PC
2		
3		
4		
5		
6		
7		

```
MOV Rd, Rs1
```

Type	15 14	13 12	11 10 9 8	7 6	5	4 3 2 1 0
Load Reg	XX	Rs1	XXXX	Rd	X	OpCode

Micro-Instructions Control: Branch / Jump Absolute Immediate (00101)

5

Bit	Operation
16	Stack Pointer Increment / Decrement (0: decrease / 1: increase)
15	Stack Pointer Count
14	RAM Address Select (0: Address register / 1: Stack pointer)
13	Use Rd as Source
12	Lower Address Register Load
11	Mem Write
10	Mem Read
9	PC Out
8	PC Load
7	PC advance
6	Program Register H Write to Bus (parameter)
5	Register File Input Enable
4	Register File Output Enable
3	Register File Output Select (parameter – 0 -> Rs1, 1 -> Rs2)
2	ALU output enable
1	ALU load register A
0	ALU load register B

Cycle #	Instruction	Explanation
0	0x1_40 01_0100_0000	Write H register to bus and PC (conditionally) reads
1	0x0_80 00_1000_0000	PC advance
2		
3		
4		
5		
6		
7		

C2	C1	C0	Condition	Low instruction register	Assembly
0	0	0	JMP	05	JMP
0	0	1	N	25	BMI – Branch on Minus
0	1	0	C	45	BCS – Branch Carry Set
0	1	1	Z	65	BEQ – Branch Equal
1	0	0	Never	85	NOP
1	0	1	!N	A5	BPL – Branch on Plus
1	1	0	!C	C5	BCC – Branch Carry Clear
1	1	1	!Z	E5	BNE

Type	15 14	13 12	11 10 9 8	7 6 5	4 3 2 1 0
Branch	Immediate (8-bit)			C2 C1 C0	OpCode

Micro-Instructions Control: ALU immediate (4-bit) (00110)

Bit	Operation
16	Stack Pointer Increment / Decrement (0: decrease / 1: increase)
15	Stack Pointer Count
14	RAM Address Select (0: Address register / 1: Stack pointer)
13	Use Rd as Source
12	Lower Address Register Load
11	Mem Write
10	Mem Read
9	PC Out
8	PC Load
7	PC advance
6	Program Register H Write to Bus (parameter)
5	Register File Input Enable
4	Register File Output Enable
3	Register File Output Select (parameter – 0 -> Rs1, 1 -> Rs2)
2	ALU output enable
1	ALU load register A
0	ALU load register B

Cycle #	Instruction	Explanation
0	10_0000_0001_0010 = 2012	Load Rd to reg A
1	0100_0001 = 0x0041	Load 4 bit imm to reg B
2	0010_0100 = 0x0024	Write ALU output to Rd
3	1000_0000 = 0x0080	Increment PC
4		
5		
6		
7		

```
ADD Rd, #0x00
```

Type	15 14 13 12	11 10 9 8	7 6	5	4 3 2 1 0
ALU	4-bit imm.	S2 S1 S0 SF (Function sel)	Rd	X	OpCode

Micro-Instructions Control: Branch / Jump Relative Immediate (00111)

Bit	Operation
16	Stack Pointer Increment / Decrement (0: decrease / 1: increase)
15	Stack Pointer Count
14	RAM Address Select (0: Address register / 1: Stack pointer)
13	Use Rd as Source
12	Lower Address Register Load
11	Mem Write
10	Mem Read
9	PC Out
8	PC Load
7	PC advance
6	Program Register H Write to Bus (parameter)
5	Register File Input Enable
4	Register File Output Enable
3	Register File Output Select (parameter – 0 -> Rs1, 1 -> Rs2)
2	ALU output enable
1	ALU load register A
0	ALU load register B

Cycle #	Instruction	Explanation
0	10_0000_0010 0x2_02	Load PC into reg A
1	00_0100_0001 0x0_41	Load immediate into reg B
2	01_0000_0100 0x1_04	Output ALU val, load PC
3	00_1000_0000 0x0_80	Advance PC
4		
5		
6		
7		

C2	C1	C0	Condition	OpCode	Assembly
0	0	0	JMP	07	JMP
0	0	1	N	27	BMI – Branch on Minus
0	1	0	C	47	BCS – Branch Carry Set
0	1	1	Z	67	BEQ – Branch Equal
1	0	0	Never	87	NOP
1	0	1	!N	A7	BPL – Branch on Plus
1	1	0	!C	C7	BCC – Branch Carry Clear
1	1	1	!Z	E7	BNE

Type	15 14	13 12	11 10 9 8	7 6 5	4 3 2 1 0
Branch	Immediate (8-bit)			C2 C1 C0	OpCode

Micro-Instructions Control: OUT (01000) (Display ASCII or Integer)

Bit	Operation
16	Stack Pointer Increment / Decrement (0: decrease / 1: increase)
15	Stack Pointer Count
14	RAM Address Select (0: Address register / 1: Stack pointer)
13	Use Rd as Source
12	Lower Address Register Load
11	Mem Write
10	Mem Read
9	PC Out
8	PC Load
7	PC advance
6	Program Register H Write to Bus (parameter)
5	Register File Input Enable
4	Register File Output Enable
3	Register File Output Select (parameter – 0 -> Rs1, 1 -> Rs2)
2	ALU output enable
1	ALU load register A
0	ALU load register B

Cycle #	Instruction	Explanation
0	0000_0000 = 0x00	Wait for H register to be loaded (need value of Rs1)
1	0001_0000 = 0x10	Write Rs1 to the bus (when opcode 01000 is detected, the value on the bus is read by the OUT register to the display)
2	1000_0000 = 0x80	Advance PC
3		
4		

```
OUT Rs1, A
```

Type	15 14	13 12	11 10 9 8	7 6	5	4 3 2 1 0
Out	XX	Rs1	XXXX	XX	I=1 A=0	OpCode

Micro-Instructions Control: LMA (Load Memory Absolute) (01001)

Bit	Operation
16	Stack Pointer Increment / Decrement (0: decrease / 1: increase)
15	Stack Pointer Count
14	RAM Address Select (0: Address register / 1: Stack pointer)
13	Use Rd as Source
12	Lower Address Register Load
11	Mem Write
10	Mem Read
9	PC Out
8	PC Load
7	PC advance
6	Program Register H Write to Bus (parameter)
5	Register File Input Enable
4	Register File Output Enable
3	Register File Output Select (parameter – 0 -> Rs1, 1 -> Rs2)
2	ALU output enable
1	ALU load register A
0	ALU load register B

Cycle #	Instruction	Explanation
0	1_0000_0100_0000 = 1040	Load 8-bit address to address register
1	0_0100_0010_0000 = 0420	Read byte from RAM and store in Rd
2	0_0000_1000_0000 = 0080	Increment PC
3		
4		
5		
6		
7		

```
LMA Rd, #Addr
```

Type	15 14 13 12 11 10 9 8	7 6	5	4 3 2 1 0
Load Reg	8-bit (LSBs) Address	Rd	X	OpCode

Micro-Instructions Control: SMA (Store Memory Absolute) (01010)

Bit	Operation
16	Stack Pointer Increment / Decrement (0: decrease / 1: increase)
15	Stack Pointer Count
14	RAM Address Select (0: Address register / 1: Stack pointer)
13	Use Rd as Source
12	Lower Address Register Load
11	Mem Write
10	Mem Read
9	PC Out
8	PC Load
7	PC advance
6	Program Register H Write to Bus (parameter)
5	Register File Input Enable
4	Register File Output Enable
3	Register File Output Select (parameter – 0 -> Rs1, 1 -> Rs2)
2	ALU output enable
1	ALU load register A
0	ALU load register B

Cycle #	Instruction	Explanation
0	01_0000_0100_0000 = 1040	Load 8-bit address to address register
1	10_1000_0001_0000 = 2810	Store byte to RAM from Rd
2	00_0000_1000_0000 = 0080	Increment PC
3		
4		
5		
6		
7		

A

```
SMA Rd, #Addr
```

Type	15 14 13 12 11 10 9 8	7 6	5	4 3 2 1 0
Load Reg	8-bit (LSBs) Address	Rd	X	OpCode

Micro-Instructions Control: LMR (Load Memory Register) (01011)

B

Bit	Operation
16	Stack Pointer Increment / Decrement (0: decrease / 1: increase)
15	Stack Pointer Count
14	RAM Address Select (0: Address register / 1: Stack pointer)
13	Use Rd as Source
12	Lower Address Register Load
11	Mem Write
10	Mem Read
9	PC Out
8	PC Load
7	PC advance
6	Program Register H Write to Bus (parameter)
5	Register File Input Enable
4	Register File Output Enable
3	Register File Output Select (parameter – 0 -> Rs1, 1 -> Rs2)
2	ALU output enable
1	ALU load register A
0	ALU load register B

Cycle #	Instruction	Explanation
0	1_0000_0001_0000 = 1010	Load Rs1 to Address Register
1	0_0100_0010_0000 = 0420	Load value at the address into Rd
2	0_0000_1000_0000 = 0080	Increment PC
3		
4		
5		
6		
7		

`LMR Rd, Rs` Effect: Rd = *Rs

Type	15 14	13 12	11 10 9 8	7 6	5	4 3 2 1 0
Load Reg	XX	Rs1	XXXX	Rd	X	OpCode

Micro-Instructions Control: SMR (Store Memory Register) (01100)

Bit	Operation
16	Stack Pointer Increment / Decrement (0: decrease / 1: increase)
15	Stack Pointer Count
14	RAM Address Select (0: Address register / 1: Stack pointer)
13	Use Rd as Source
12	Lower Address Register Load
11	Mem Write
10	Mem Read
9	PC Out
8	PC Load
7	PC advance
6	Program Register H Write to Bus (parameter)
5	Register File Input Enable
4	Register File Output Enable
3	Register File Output Select (parameter – 0 -> Rs1, 1 -> Rs2)
2	ALU output enable
1	ALU load register A
0	ALU load register B

Cycle #	Instruction	Explanation
0	01_0000_0001_0000 = 1010	Load Rs1 to Address Register
1	10_1000_0001_0000 = 2810	Store value at Rd to RAM
2	00_0000_1000_0000 = 0080	Increment PC
3		
4		
5		
6		
7		

```
SMR Rd_data, Rs1_addr
```

Type	15 14	13 12	11 10 9 8	7 6	5	4 3 2 1 0
Load Reg	XX	Rs1	XXXX	Rd	X	OpCode

Micro-Instructions Control: OUT immediate (01101)

D

Bit	Operation
16	Stack Pointer Increment / Decrement (0: decrease / 1: increase)
15	Stack Pointer Count
14	RAM Address Select (0: Address register / 1: Stack pointer)
13	Use Rd as Source
12	Lower Address Register Load
11	Mem Write
10	Mem Read
9	PC Out
8	PC Load
7	PC advance
6	Program Register H Write to Bus (parameter)
5	Register File Input Enable
4	Register File Output Enable
3	Register File Output Select (parameter – 0 -> Rs1, 1 -> Rs2)
2	ALU output enable
1	ALU load register A
0	ALU load register B

Cycle #	Instruction	Explanation
0	0000_0000 = 0x00	Wait for H instruction register to be loaded
1	0100_0000 = 0x40	Write program register H to the bus
2	1000_0000 = 0x80	Advance PC
3		
4		

```
OUT Rs1, A
```

Type	15 14	13 12	11 10 9 8	7 6	5	4 3 2 1 0
Out	8-bit immediate			XX	I=1 A=0	OpCode

Micro-Instructions Control: PUSH (01110)

Bit	Operation
16	Stack Pointer Increment / Decrement (0: decrease / 1: increase)
15	Stack Pointer Count
14	RAM Address Select (0: Address register / 1: Stack pointer)
13	Use Rd as Source
12	Lower Address Register Load
11	Mem Write
10	Mem Read
9	PC Out
8	PC Load
7	PC advance
6	Program Register H Write to Bus (parameter)
5	Register File Input Enable
4	Register File Output Enable
3	Register File Output Select (parameter – 0 -> Rs1, 1 -> Rs2)
2	ALU output enable
1	ALU load register A
0	ALU load register B

Cycle #	Instruction	Explanation
0	0_0100_1000_0001_0000 = 0x04810	Store Rs1 in the stack
1	1_1000_0000_1000_0000 = 0x18080	Increment stack pointer (and increment PC)
2		
3		
4		
5		
6		
7		

PUSH Rs1

Type	15 14	13 12	11 10 9 8	7 6	5	4 3 2 1 0
Load Reg	XX	Rs1	XXXX	XX	X	OpCode

Micro-Instructions Control: POP (1111)

F

Bit	Operation
16	Stack Pointer Increment / Decrement (0: decrease / 1: increase)
15	Stack Pointer Count
14	RAM Address Select (0: Address register / 1: Stack pointer)
13	Use Rd as Source
12	Lower Address Register Load
11	Mem Write
10	Mem Read
9	PC Out
8	PC Load
7	PC advance
6	Program Register H Write to Bus (parameter)
5	Register File Input Enable
4	Register File Output Enable
3	Register File Output Select (parameter – 0 -> Rs1, 1 -> Rs2)
2	ALU output enable
1	ALU load register A
0	ALU load register B

Cycle #	Instruction	Explanation
0	0_0000_0000_0000_0000 = 0x00000	Wait for next clock
1	0_1000_0000_0000_0000 = 0x08000	Decrement stack pointer
2	0_0100_0100_0010_0000 = 0x04420	Load value from stack to Rd
3	0_0000_0000_1000_0000 = 0x00080	Increment PC
4		
5		
6		
7		

POP Rd

Type	15 14	13 12	11 10 9 8	7 6	5 4	3 2 1 0
Load Reg	XX	XX	XXXX	Rd	X X	OpCode

Micro-Instructions Control: JSR Jump Subroutine (10000)

Bit	Operation
16	Stack Pointer Increment / Decrement (0: decrease / 1: increase)
15	Stack Pointer Count
14	RAM Address Select (0: Address register / 1: Stack pointer)
13	Use Rd as Source
12	Lower Address Register Load
11	Mem Write
10	Mem Read
9	PC Out
8	PC Load
7	PC advance
6	Program Register H Write to Bus (parameter)
5	Register File Input Enable
4	Register File Output Enable
3	Register File Output Select (parameter – 0 -> Rs1, 1 -> Rs2)
2	ALU output enable
1	ALU load register A
0	ALU load register B

Cycle #	Instruction	Explanation
0	0_0100_1010_0000_0000 = 0x04A00	Write PC to stack
1	1_1000_0000_0000_0000 = 0x18000	Increase stack pointer
2	0_0000_0001_0100_0000 = 0x00140	Jump to immediate address
3	0_0000_0000_1000_0000 = 0x00080	Advance PC
4		
5		
6		
7		

JSR #Imm

Type	15 14 13 12 11 10 9 8	7 6	5 4	3 2 1 0
Load Reg	8-bit Immediate (Program Address)	XX	X X	OpCode

Micro-Instructions Control: RTS Return from Subroutine (10001)

Bit	Operation
16	Stack Pointer Increment / Decrement (0: decrease / 1: increase)
15	Stack Pointer Count
14	RAM Address Select (0: Address register / 1: Stack pointer)
13	Use Rd as Source
12	Lower Address Register Load
11	Mem Write
10	Mem Read
9	PC Out
8	PC Load
7	PC advance
6	Program Register H Write to Bus (parameter)
5	Register File Input Enable
4	Register File Output Enable
3	Register File Output Select (parameter – 0 -> Rs1, 1 -> Rs2)
2	ALU output enable
1	ALU load register A
0	ALU load register B

Cycle #	Instruction	Explanation
0	0_0000_0000_0000_0000 = 0x00000	Wait a clock cycle
1	0_1000_0000_0000_0000 = 0x08000	Decrement stack pointer
2	0_0100_0101_0000_0000 = 0x04500	Load program address on the stack to the program counter
3	0_0000_0000_1000_0000 = 0x00080	Advance PC
4		
5		
6		
7	RTS	

Type	15 14 13 12 11 10 9 8	7 6	5 4	3 2 1 0
Load Reg	XXXX XXXX	XX	X X	OpCode

Micro-Instructions Control: Test (10010)

Bit	Operation
16	Stack Pointer Increment / Decrement (0: decrease / 1: increase)
15	Stack Pointer Count
14	RAM Address Select (0: Address register / 1: Stack pointer)
13	Use Rd as Source
12	Lower Address Register Load
11	Mem Write
10	Mem Read
9	PC Out
8	PC Load
7	PC advance
6	Program Register H Write to Bus (parameter)
5	Register File Input Enable
4	Register File Output Enable
3	Register File Output Select (parameter – 0 -> Rs1, 1 -> Rs2)
2	ALU output enable
1	ALU load register A
0	ALU load register B

Cycle #	Instruction	Explanation
0	0001_0010 = 0x12	Load Rs1 to reg A
1	0001_1001 = 0x19	Load Rs2 to reg B
2	0000_0100 = 0x04	Output ALU value on bus (but don't save in register)
3	1000_0000 = 0x80	Increment PC
4		
5		
6		
7		

```
TST Rs1, Rs2
```

Type	15 14	13 12	11 10 9 8	7 6	5 4	3 2 1 0
ALU	Rs2	Rs1	S2 S1 S0 SF (Function sel)	Rd	XX	OpCode

Micro-Instructions Control: Test immediate (10011)

Bit	Operation
16	Stack Pointer Increment / Decrement (0: decrease / 1: increase)
15	Stack Pointer Count
14	RAM Address Select (0: Address register / 1: Stack pointer)
13	Use Rd as Source
12	Lower Address Register Load
11	Mem Write
10	Mem Read
9	PC Out
8	PC Load
7	PC advance
6	Program Register H Write to Bus (parameter)
5	Register File Input Enable
4	Register File Output Enable
3	Register File Output Select (parameter – 0 -> Rs1, 1 -> Rs2)
2	ALU output enable
1	ALU load register A
0	ALU load register B

Cycle #	Instruction	Explanation
0	10_0000_0001_0010 = 0x2012	Load Rd to reg A
1	0100_0001 = 0x0041	Load 4 bit imm to reg B
2	0000_0100 = 0x0004	Write ALU output to bus (but not to register)
3	1000_0000 = 0x80	Increment PC
4		
5		
6		
7		

```
TST Rd, #0x00
```

Type	15 14 13 12	11 10 9 8	7 6	5 4	3 2 1 0
ALU	4-bit imm.	S2 S1 S0 SF (Function sel)	Rd	XX	OpCode

ALU

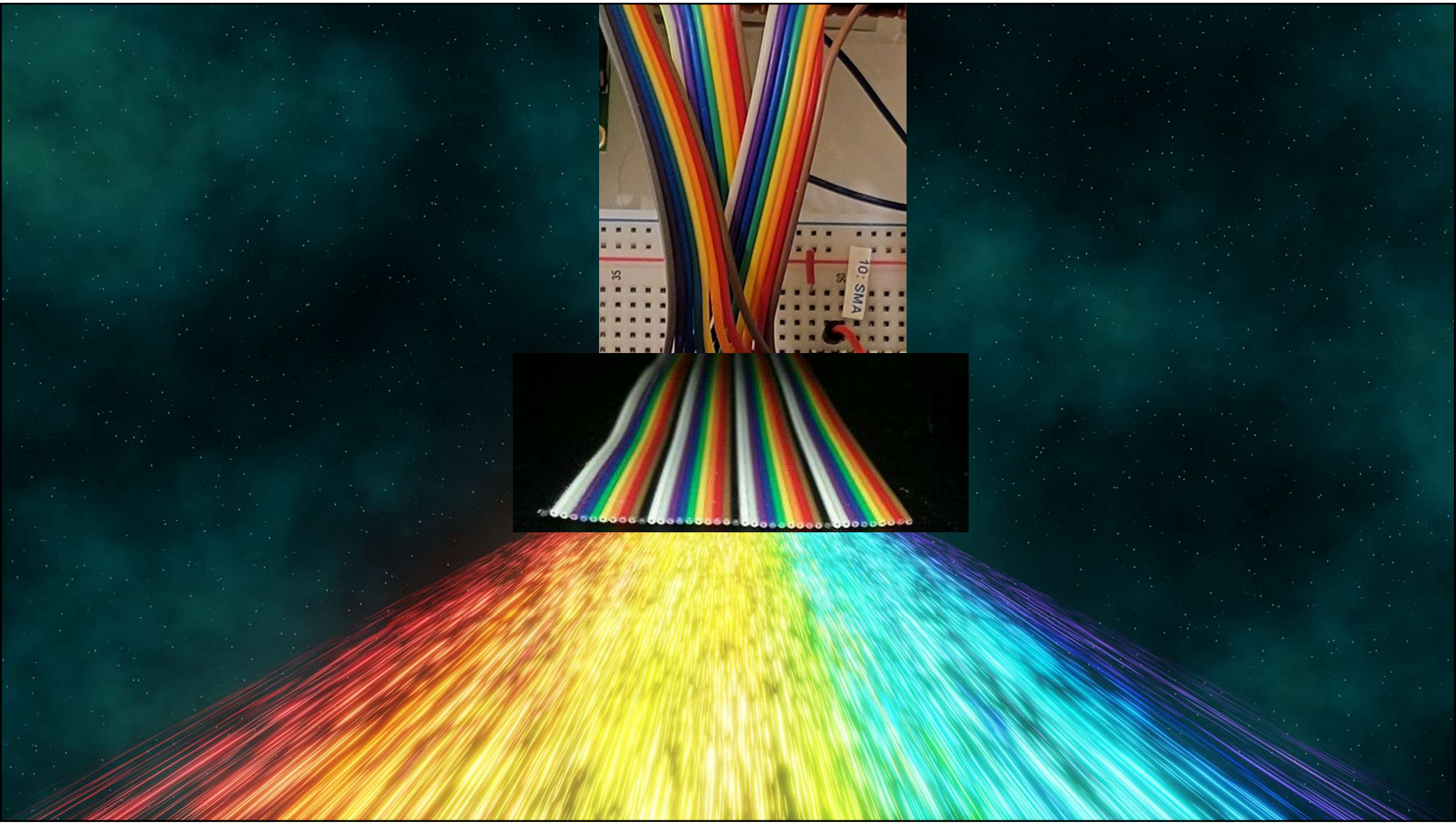
Op Select	Special Func. = 0	Special Func. = 1	Flag in	Flags Out
0	A OR B	A OR B	-	Z, N
1	A AND B	A AND B	-	Z, N
2	B	~B	-	Z, N
3	A + B	A – B	-	Z, N, Co
4	Shift Left A A[0]=0	Rotate left A A[0]=Cin	Cin (Rotate)	Z, N=A[6], Co=A[7]
5	Shift Right A	Arithmetic S.R. A	-	Z, N, Co=A[0]
6				
7				

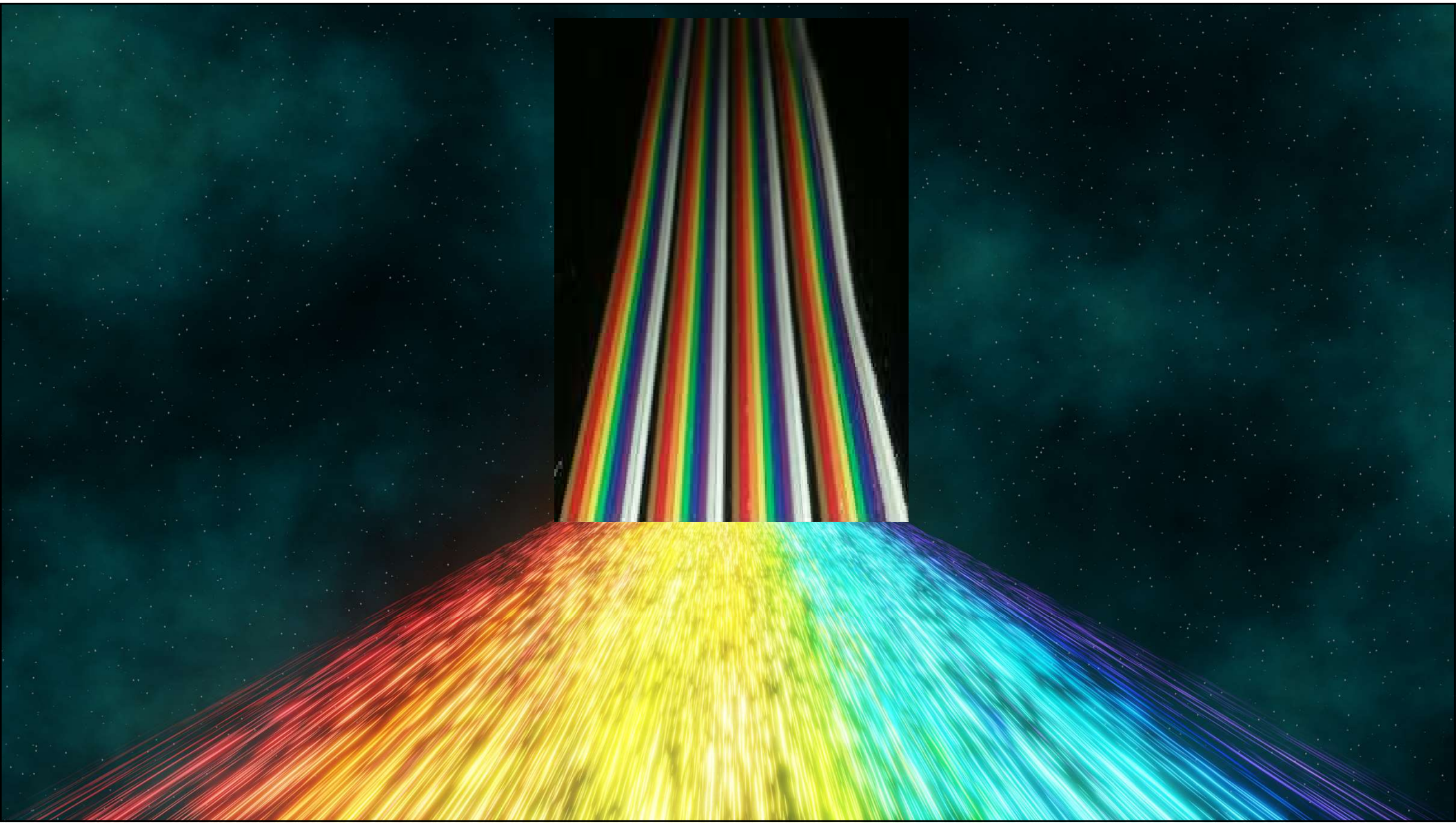
Proposal	# of IC slots	Description	Op Code
Move A, B Regs	4x 74LS161	Reduce interface to one data bus instead of three (A, B, C)	No change

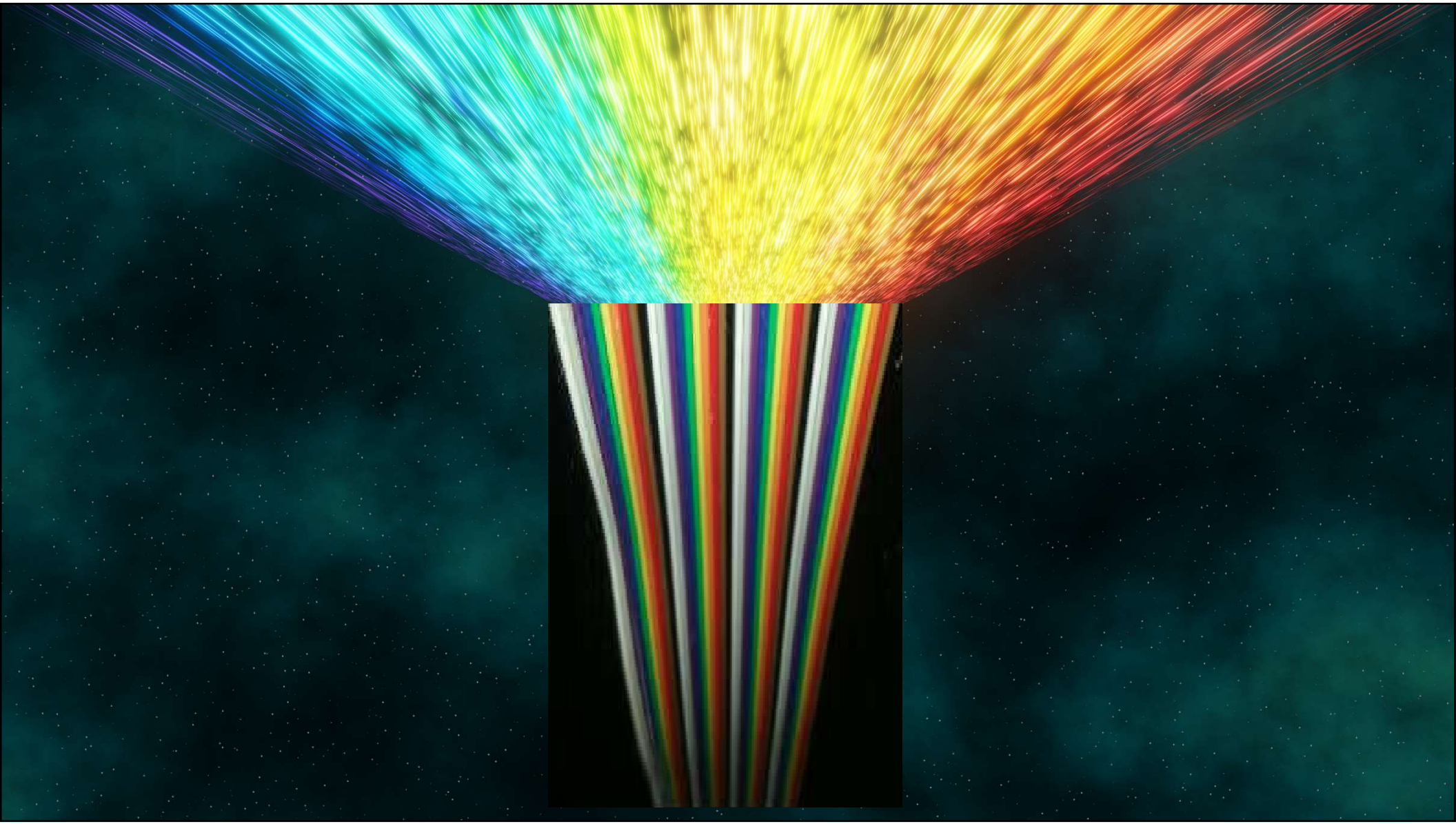
Proposals for the additional operations

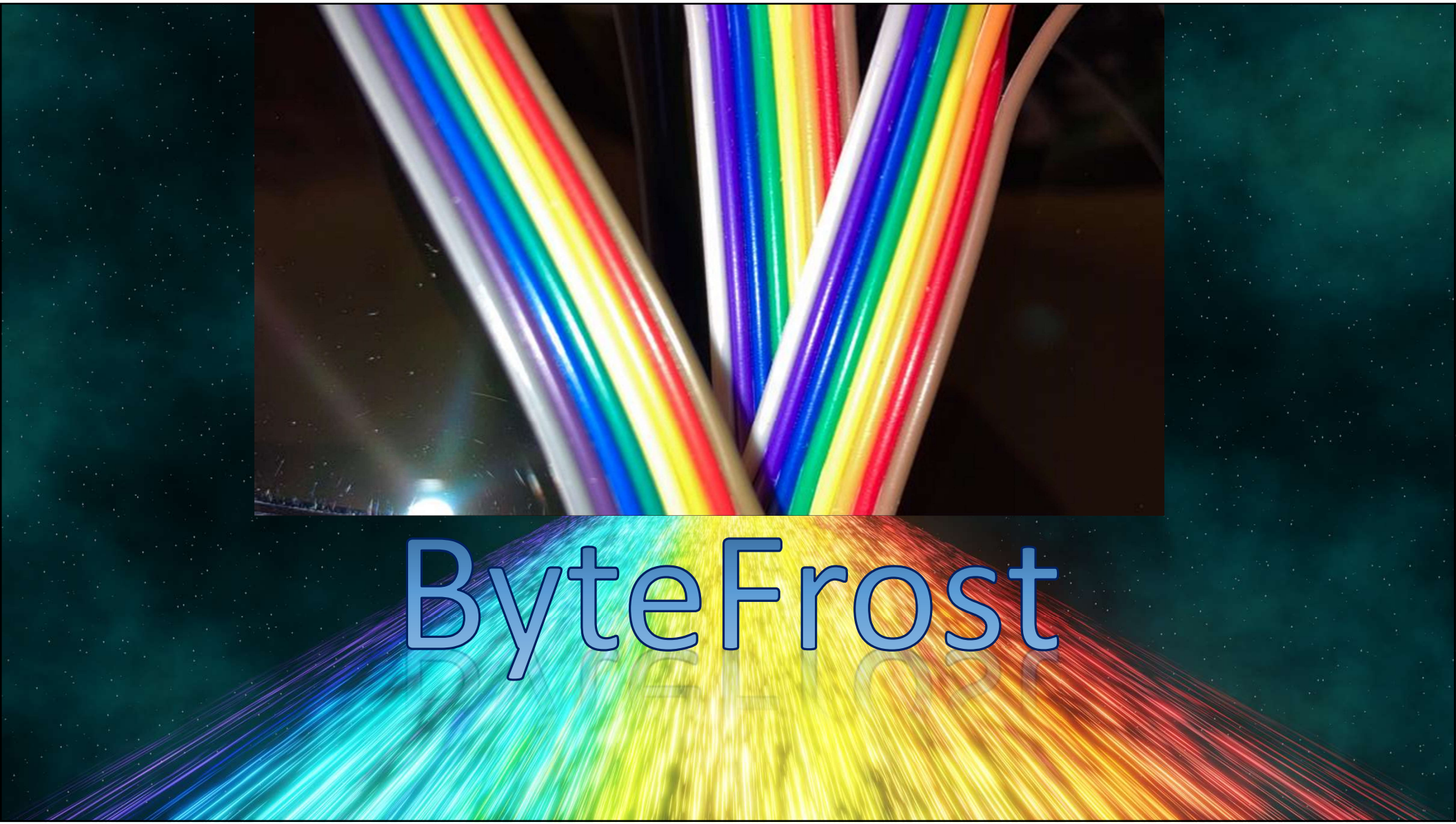
- How to chain Shift Right of two bytes?

And
the
Name
…


35
85
10: SMA






ByteFrost