6502 Subset Curriculum

James Lepone

Problem

- Currently Two Processing Classes
  - Processors II
  - C & Data Structures (Assembly Portion)

Problem

- Processors and Data Structures are similar, but not connected
- Desire to connect the classes
  - Prior understanding
  - Complex programs
  - More interest

Solution

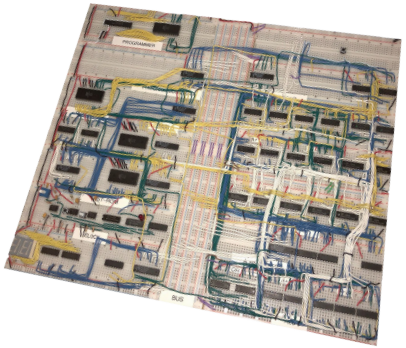
- Create a new curriculum for Processors II
- Based on 6502 from C & Data Structures
- Students will:
  - Build the 6502 subset - Milestones
  - Create/run programs on said processor
  - Expand (Opcodes & More Circuitry)

Mentor

- Mr. Ben Isecke
  - Teacher of Computer Science at BCA
  - Runs the current Processors I & Processors II Classes
  - Works closely with Mr. Respass, who teaches C & Data Structures

Stages of Project Development

- Stage One: Research
  - Physical model of the Redeemer College processor
  - Research 6502 and determine a proper subset to be built

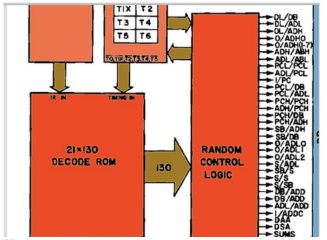


What is the Random Control Logic in the 6502?

1

I am currently developing a subset of the 6502 in LogiSim and at the current stage I am determining which parts to implement and what can be cut out. One of my main resources is Hanson's Block Diagram.

I am currently trying to determine how exactly the instructions are decoded into the control lines. In the diagram below, there are two parts, the Decode ROM and the Random Control Logic.



1 Answer

active oldest votes

3

I'm on the outskirts of my knowledge here, but my understanding is that the PLA decode ROM outputs its 130 control signals as a function of opcode and cycle, and the random logic is a purely functional unit that takes the PLA output as input in order to control the rest of the chip. I think you could combine the two into a single ROM; from looking at the die shot the random logic is about twice as large as the PLA so my guess would be that time/cost considerations, possibly including intelligent task subdivision and almost certainly including a calculation of debugging time as the 6502 was laid out literally by hand, using pen and ruler, led to the combined approach.

share edit flag

answered Nov 8 '17 at 14:25

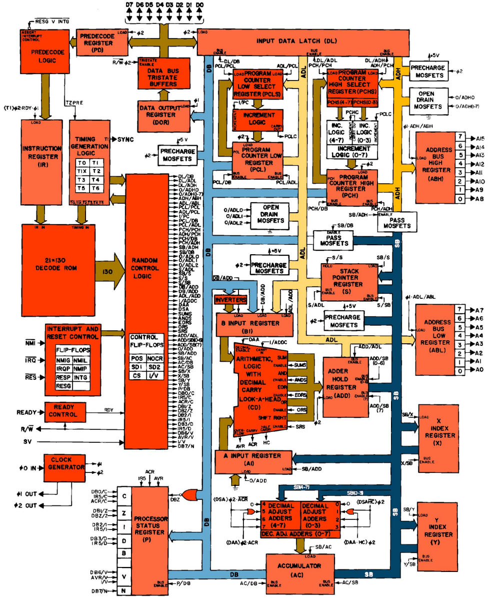
Tommy
90.2k ● 11 ● 151 ● 176

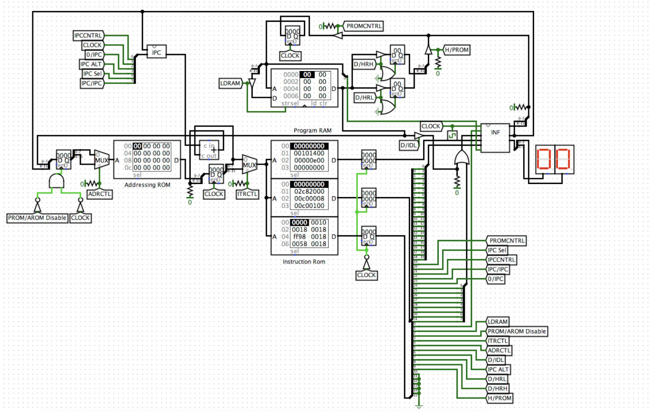
Thanks so much, that really helps. – James Lepone Nov 9 '17 at 15:05

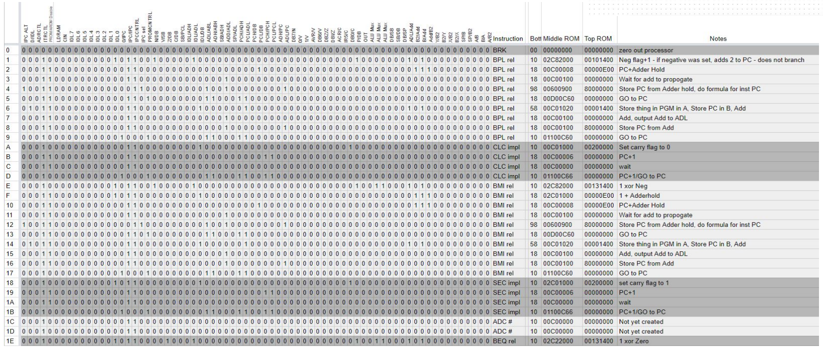
add a comment

Stages of Project Development

- Stage Two: Building & Programming







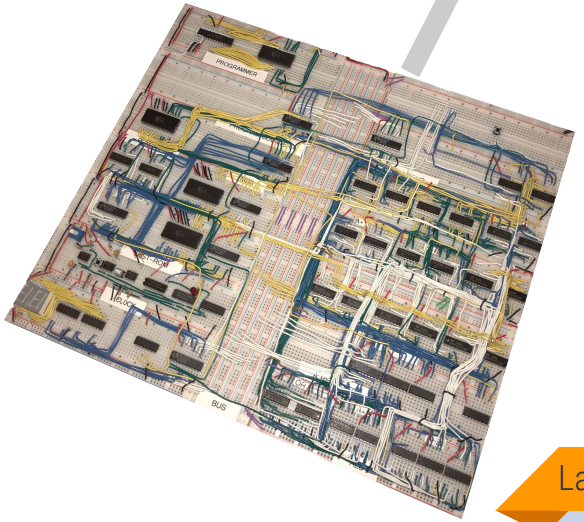
	Instruction	Bott	Middle ROM	Top ROM	Notes
0	BRK	00	00000000	00000000	zero out processor
1	BPL rel	10	02C82000	00101400	Neg flag+1 - if negative was set, adds 2 to PC - does not branch
2	BPL rel	18	00C00008	00000E00	PC+Adder Hold
3	BPL rel	18	00C00100	00000000	Wait for add to propogate
4	BPL rel	98	00600900	80000000	Store PC from Adder hold, do formula for inst PC
5	BPL rel	18	00D00C60	00000000	GO to PC
6	BPL rel	58	00C01020	00001400	Store thing in PGM in A, Store PC in B, Add
7	BPL rel	18	00C00100	00000000	Add, output Add to ADL
8	BPL rel	18	00C00100	80000000	Store PC from Add
9	BPL rel	10	01100C60	00000000	GO to PC
A	CLC impl	10	00C01000	00200000	Set carry flag to 0
B	CLC impl	18	00C00006	00000000	PC+1
C	CLC impl	18	00C00000	00000000	wait
D	CLC impl	10	01100C66	00000000	PC+1/GO to PC
E	BMI rel	10	02C82000	00131400	1 xor Neg
F	BMI rel	18	02C01000	00000E00	1 + Adderhold
10	BMI rel	18	00C00008	00000E00	PC+Adder Hold
11	BMI rel	18	00C00100	00000000	Wait for add to propogate
12	BMI rel	98	00600900	80000000	Store PC from Adder hold, do formula for inst PC
13	BMI rel	18	00D00C60	00000000	GO to PC
14	BMI rel	58	00C01020	00001400	Store thing in PGM in A, Store PC in B, Add
15	BMI rel	18	00C00100	00000000	Add, output Add to ADL
16	BMI rel	18	00C00100	80000000	Store PC from Add
17	BMI rel	10	01100C60	00000000	GO to PC
18	SEC impl	10	02C01000	00200000	set carry flag to 1
19	SEC impl	18	00C00006	00000000	PC+1
1A	SEC impl	18	00C00000	00000000	wait
1B	SEC impl	10	01100C66	00000000	PC+1/GO to PC
1C	ADC #	10	00C00000	00000000	Not yet created
1D	ADC #	10	00C00000	00000000	Not yet created
1E	BEQ rel	10	02C22000	00131400	1 xor Zero

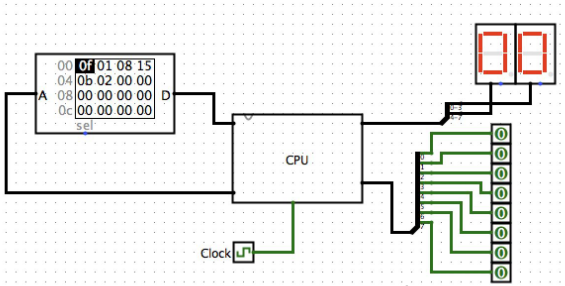
Stages of Project Development

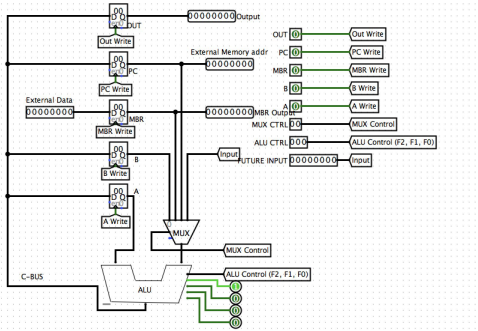
- Stage Three: Develop Curriculum
  - Create a Milestone Style assignment system for the developed subset

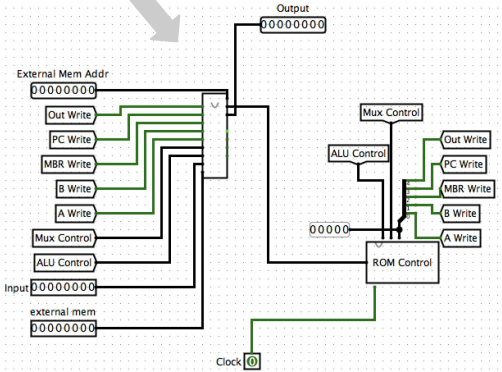
Technical Overview

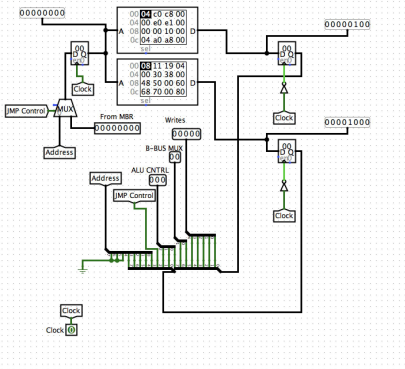
Processor Design







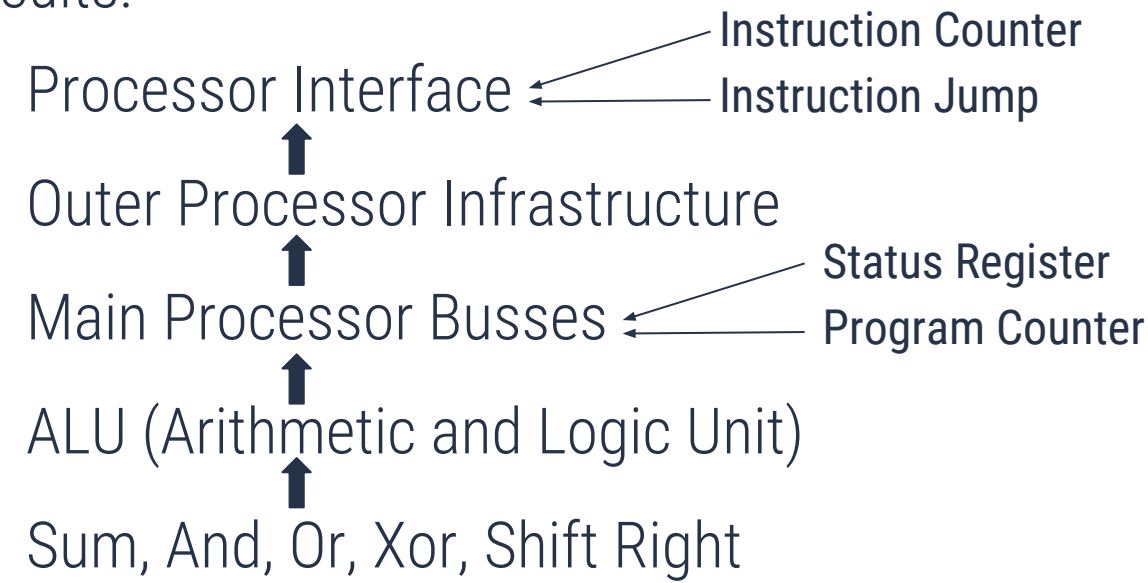




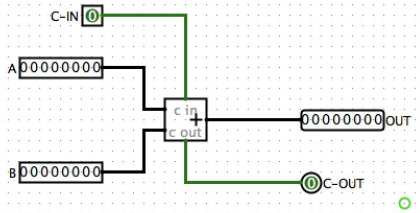
Large breadboard project can be simplified into smaller circuits made up of subcircuits

Technical Overview - Processor Design

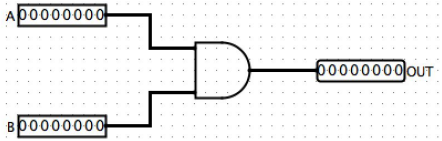
Circuits:



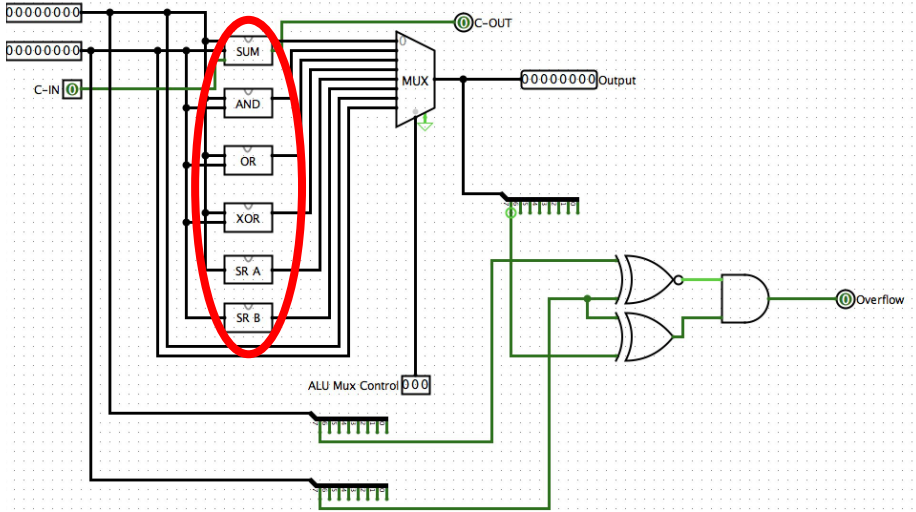
Sum



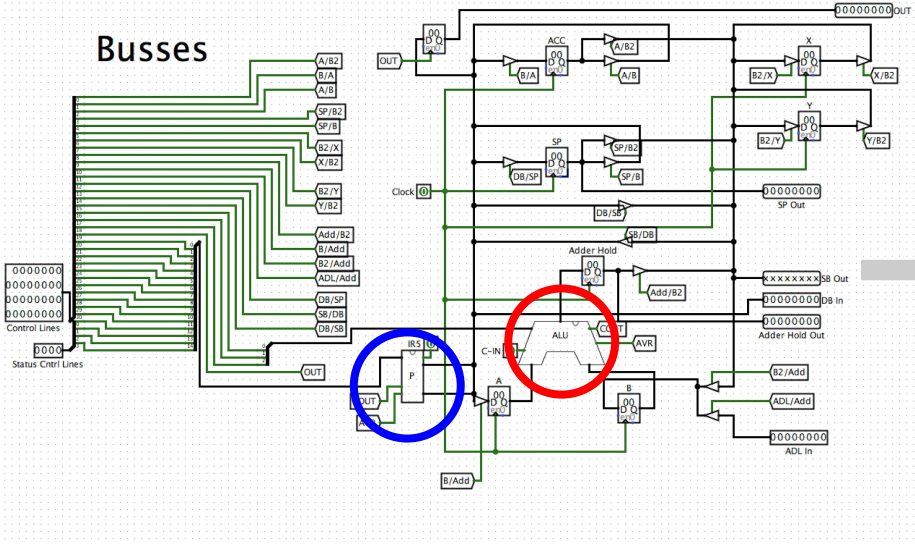
And

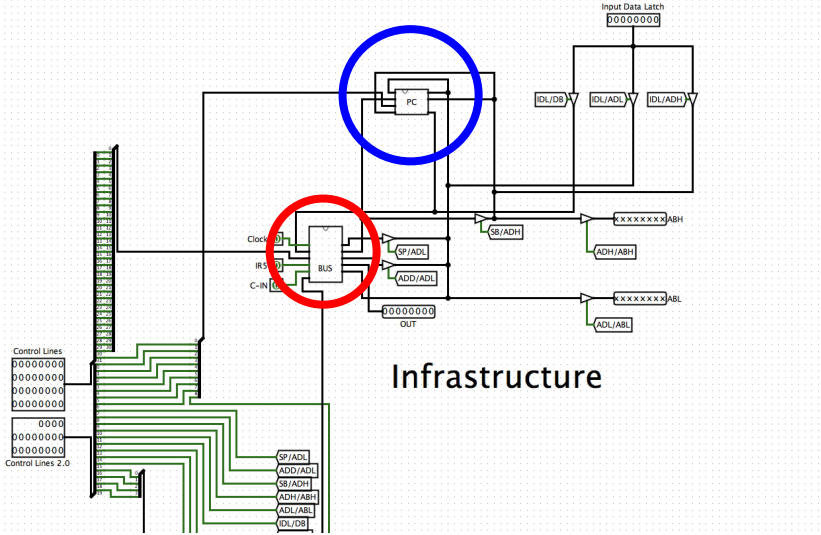


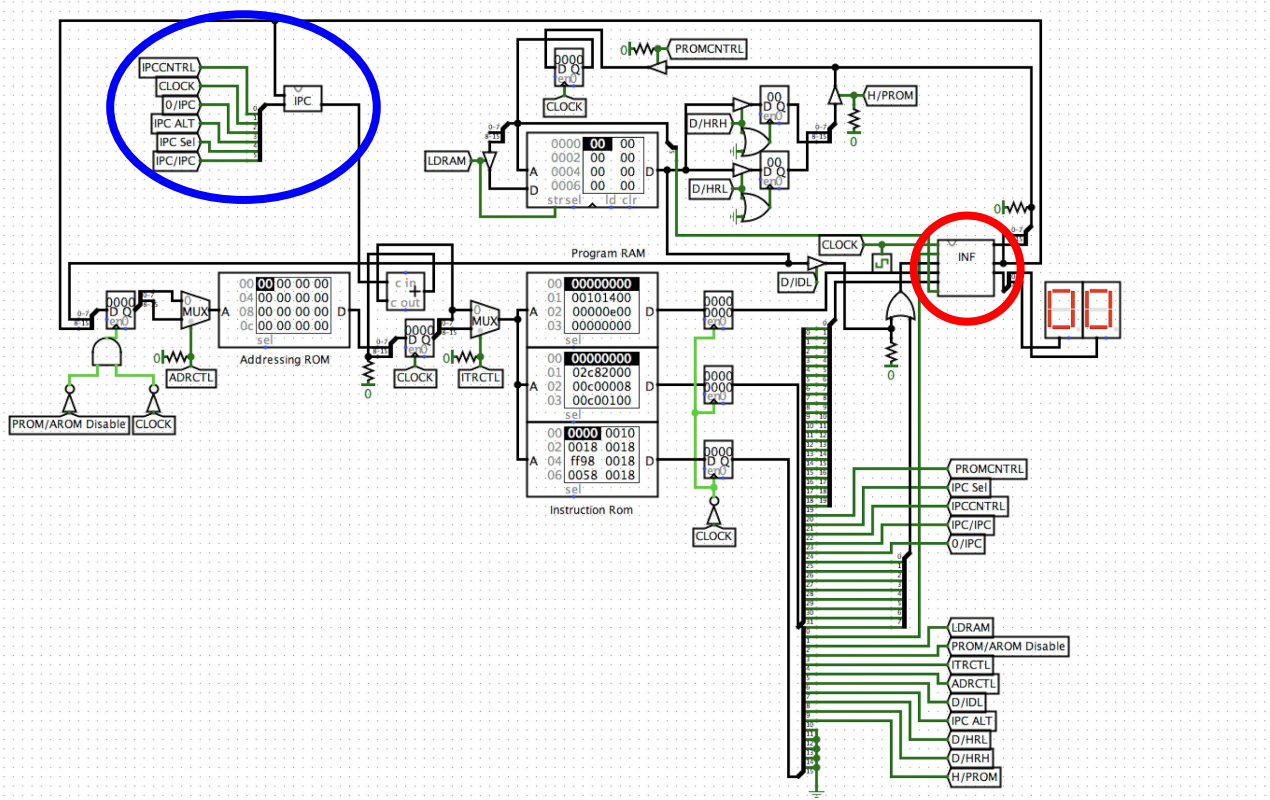
ALU



And More...

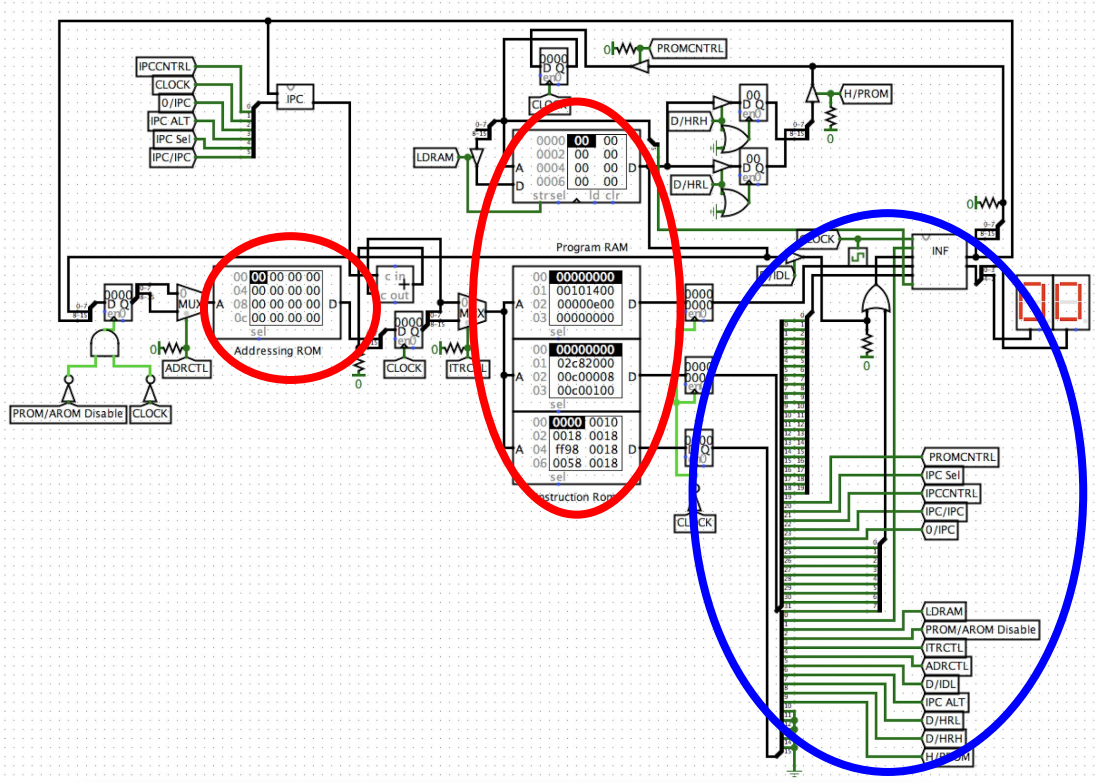

Busses
A/B2
B/A
A/B
SP/B2
SP/B
B2/X
X/B2
B2/Y
Y/B2
Add/B2
B/Add
B2/Add
ADL/Add
DB/SP
SB/DB
DB/SB
Control Lines
Status Cntrl Lines
OUT
ACC
X
Y
SP
Clock
Adder Hold
SP Out
SB Out
DB In
Adder Hold Out
ALU
C-IN
AVR
IRS
P
A
B
ADL/Add
ADL In


Input Data Latch
PC
IDL/DB
IDL/ADL
IDL/ADH
Clock
IRS
C-IN
BUS
SP/ADL
ADD/ADL
SB/ADH
ADH/ABH
ADL/ABL
OUT
Control Lines
Control Lines 2.0
Infrastructure


IPCCNTRL
CLOCK
0/IPC
IPC ALT
IPC Sel
IPC/IPC
IPC
PROMCNTRL
CLOCK
H/PROM
D/HRH
D/HRL
LDRAM
Program RAM
CLOCK
INF
D/IDL
MUX
Addressing ROM
ADRCTL
PROM/AROM Disable
CLOCK
CLOCK
ITRCTL
Instruction Rom
CLOCK
PROMCNTRL
IPC Sel
IPCCNTRL
IPC/IPC
0/IPC
LDRAM
PROM/AROM Disable
ITRCTL
ADRCTL
D/IDL
IPC ALT
D/HRL
D/HRH
H/PROM

Technical Overview

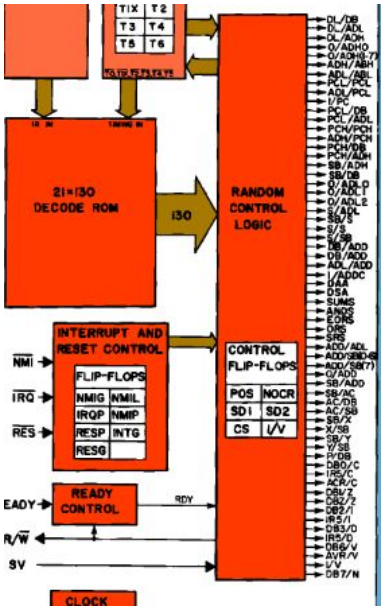
Processor Code



ROM Chips (Red), Control Lines (Blue)

Technical Overview - Processor Code

Utilized an excel spreadsheet to program these control lines.



Control Lines

Padding	71
D/IDL	70
ADRCTL	69
ITRCTL	68
PROM/AROM Di	67
LDRAM	66
IR5	65
IDL 7	64
IDL 6	63
IDL 5	62
IDL 4	61
IDL 3	60
IDL 2	59
IDL 1	58
IDL 0	57
N/DB	56
V/SB	55
Z/DB	54
C/DB	53
IPC/DB	52
IPC/IPC	51
IPC/ADL	50
ADH/IPC	49
ADL/IPC	48
SB/PCL	47
IDL/ADH	46
IDL/ADL	45
IDL/DB	44
ADL/ABL	43
ADH/ABH	42
SB/ADH	41

Addressing

Mem	Binary	Hex	Command
0	0	0	BRK impl
1	0	0	
2	0	0	
3	0	0	
4	0	0	
5	0	0	
6	0	0	
7	0	0	
8	0	0	
9	0	0	
A	0	0	
B	0	0	
C	0	0	
D	0	0	
E	0	0	
F	0	0	
10	1	1	BPL rel
11	0	0	
12	0	0	
13	0	0	
14	0	0	
15	0	0	
16	0	0	
17	0	0	
18	111	7	CLC impl
19	0	0	
1A	0	0	
1B	0	0	
1C	0	0	
1D	0	0	

Technical Overview - Processor Code

Instruction

	IPC ALT	D/IDL	ADRCTL	ITRCTL	PROM/AROM Disable	LDRAM	CIN	IDL 7	IDL 6	IDL 5	IDL 4	IDL 3	IDL 2	IDL 1	IDL 0	0/IPC	IPC/IPC	IPCCNTRL	IPC sel	PROMCNTRL	N/DB	V/SB	Z/DB	C/DB	SB/PCL	IDL/ADH	IDL/ADL	IDL/DB	ADL/ABL	ADH/ABH	SB/ADH	ADD/ADL	SP/ADL	PCH/ADH	PCL/ADL	PCH/DB	PCL/DB	PCH/PCH	PCL/PCL	ADH/PC	ADL/PC	DB7/N	0/V	1/V	AVR/V	DB6/V	DBZ/Z	DBI/Z	ACR/C	IR5/C	DB0/C	P/DB	OUT	ALU Mux	ALU Mux	ALU Mux	DB/SB	SB/DB	DB/SP	ADL/Add	B2/Add	B/Add	Add/B2	Y/B2	B2/Y	X/B2	B2/X	SP/B	SP/B2	A/B	B/A	A/B2	Instruction	Bott	Middle ROM	Top ROM	Notes
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	BRK	00	00000000	00000000	zero out processor
1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	BPL rel	10	02C82000	00101400	Neg flag+1 - if negative was set, adds 2 to PC - does not branch
2	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	BPL rel	18	00C00008	00000E00	PC+Adder Hold
3	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	BPL rel	18	00C00100	00000000	Wait for add to propogate
4	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	BPL rel	98	00600900	80000000	Store PC from Adder hold, do formula for inst PC
5	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	1	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	BPL rel	18	00D00C60	00000000	GO to PC
6	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	BPL rel	58	00C01020	00001400	Store thing in PGM in A, Store PC in B, Add
7	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	BPL rel	18	00C00100	00000000	Add, output Add to ADL
8	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	BPL rel	18	00C00100	80000000	Store PC from Add
9	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	1	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	BPL rel	10	01100C60	00000000	GO to PC
A	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CLC impl	10	00C01000	00200000	Set carry flag to 0
B	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CLC impl	18	00C00006	00000000	PC+1
C	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CLC impl	18	00C00000	00000000	wait
D	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	1	1	0	0	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CLC impl	10	01100C66	00000000	PC+1/GO to PC
E	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	BMI rel	10	02C82000	00131400	1 xor Neg
F	0	0	0	1	1	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	BMI rel	18	02C01000	00000E00	1 + Adderhold
10	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	BMI rel	18	00C00008	00000E00	PC+Adder Hold
11	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	BMI rel	18	00C00100	00000000	Wait for add to propogate
12	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	BMI rel	98	00600900	80000000	Store PC from Adder hold, do formula for inst PC
13	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	1	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	BMI rel	18	00D00C60	00000000	GO to PC
14	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	BMI rel	58	00C01020	00001400	Store thing in PGM in A, Store PC in B, Add
15	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	BMI rel	18	00C00100	00000000	Add, output Add to ADL
16	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	BMI rel	18	00C00100	80000000	Store PC from Add
17	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	1	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	BMI rel	10	01100C60	00000000	GO to PC
18	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	SEC impl	10	02C01000	00200000	set carry flag to 1
19	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	SEC impl	18	00C00006	00000000	PC+1
1A	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	SEC impl	18	00C00000	00000000	wait
1B	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	1	1	0	0	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	SEC impl	10	01100C66	00000000	PC+1/GO to PC
1C	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	ADC #	10	00C00000	00000000	Not yet created
1D	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	ADC #	10	00C00000	00000000	Not yet created
1E	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	BEQ rel	10	02C22000	00131400	1 xor Zero

Milestones

- How can I get the students to build what I built?
- Small, self-guided tasks that students use as a guide towards building the processor

Milestone 1

ALU and Its Components

If the processor was a being then the ALU would be the heart, this is the part of the processor that handles all of the math, which essentially is what a processor does at its core. This means that the first step in our journey to build the 6502 processor begins by building the ALU. Note that the 6502 uses 8 bit busses, unless otherwise noted.

Step 1: The Main Components

There are 6 main operations that the 6502 ALU handles.

1. SUM - Adds two numbers together (binary addition)
2. AND - Bitwise AND of the two inputs
3. OR - Bitwise OR of the two inputs
4. XOR - Bitwise XOR of the two inputs
5. SR A - Shifts the binary (of input A) one to the right
6. SR B - Shifts the binary (of input B) one to the right

LogiSim has many of these components built already, just add the inputs and output.

Expected Output

Control 1	Control 2	Control 3	Output
0	0	0	SUM
0	0	1	AND
0	1	0	OR
0	1	1	XOR
1	0	0	SRA
1	0	1	SRB
1	1	0	A
1	1	1	B

These main components (hmm, maybe we can make these into subcircuits?) make up the ALU, along with some infrastructure built upon it.

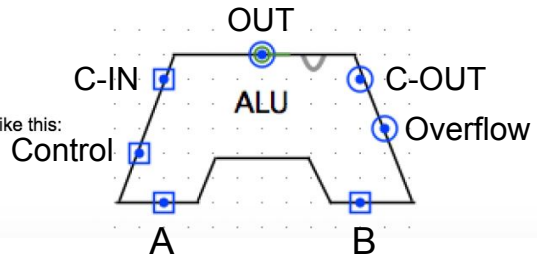
Step 2: The Infrastructure

In order to finish the ALU, one must add a way of selecting one of the functions from the rest (utilizing some control bits) and also output the carry out and overflow of the function that was used (if it applies).

Tasks For This Milestone:

1. Build the 6 main components of the ALU in seperate circuits.
2. Add these to your main ALU circuit
3. Hook these (main components) into something that can select one of them using some control bits
4. Add your inputs and outputs
5. Add circuit to detect overflow
6. Design the final ALU subcircuit.

Your final circuit should look something like this:



Future Modifications

- Simplification of Processor Design
- Developing more Opcodes
- Further development of lessons (milestones)
  - Tailor to students reactions next year

Future Potential

- Building a physical model of the processor
- Add all versions of core opcodes

HI	LO-NIBBLE															
	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00	BRK impl	ORA X,ind				ORA zpg	ASL zpg		PHP impl	ORA #	ASL A			ORA abs	ASL abs	
10	BPL rel	ORA ind,Y				ORA zpg,X	ASL zpg,X		CLC impl	ORA abs,Y				ORA abs,X	ASL abs,X	
20	JSR abs	AND X,ind			BIT zpg	AND zpg	ROL zpg		PLP impl	AND #	ROL A		BIT abs	AND abs	ROL abs	
30	BMI rel	AND ind,Y				AND zpg,X	ROL zpg,X		SEC impl	AND abs,Y				AND abs,X	ROL abs,X	
40	RTI impl	EOR X,ind				EOR zpg	LSR zpg		PHA impl	EOR #	LSR A		JMP abs	EOR abs	LSR abs	
50	BVC rel	EOR ind,Y				EOR zpg,X	LSR zpg,X		CLI impl	EOR abs,Y				EOR abs,X	LSR abs,X	
60	RTS impl	ADC X,ind				ADC zpg	ROR zpg		PLA impl	ADC #	ROR A		JMP ind	ADC abs	ROR abs	
70	BVS rel	ADC ind,Y				ADC zpg,X	ROR zpg,X		SEI impl	ADC abs,Y				ADC abs,X	ROR abs,X	
80		STA X,ind			STY zpg	STA zpg	STX zpg		DEY impl		TXA impl		STY abs	STA abs	STX abs	
90	BCC rel	STA ind,Y			STY zpg,X	STA zpg,X	STX zpg,Y		TYA impl	STA abs,Y	TXS impl			STA abs,X		
A0	LDY #	LDA X,ind	LDX #		LDY zpg	LDA zpg	LDX zpg		TAY impl	LDA #	TAX impl		LDY abs	LDA abs	LDX abs	
B0	BCS rel	LDA ind,Y			LDY zpg,X	LDA zpg,X	LDX zpg,Y		CLV impl	LDA abs,Y	TSX impl		LDY abs,X	LDA abs,X	LDX abs,Y	
C0	CPY #	CMP X,ind			CPY zpg	CMP zpg	DEC zpg		INY impl	CMP #	DEX impl		CPY abs	CMP abs	DEC abs	
D0	BNE rel	CMP ind,Y				CMP zpg,X	DEC zpg,X		CLD impl	CMP abs,Y				CMP abs,X	DEC abs,X	
E0	CPX #	SBC X,ind			CPX zpg	SBC zpg	INC zpg		INX impl	SBC #	NOP impl		CPX abs	SBC abs	INC abs	
F0	BEQ rel	SBC ind,Y				SBC zpg,X	INC zpg,X		SED impl	SBC abs,Y				SBC abs,X	INC abs,X	

Reflections

Thank You!

Thank you for your attention!

Special Thanks To:

- Mr. Isecke
- Mr. Respass
- The Stack Overflow Community
- The Bergen County Academies