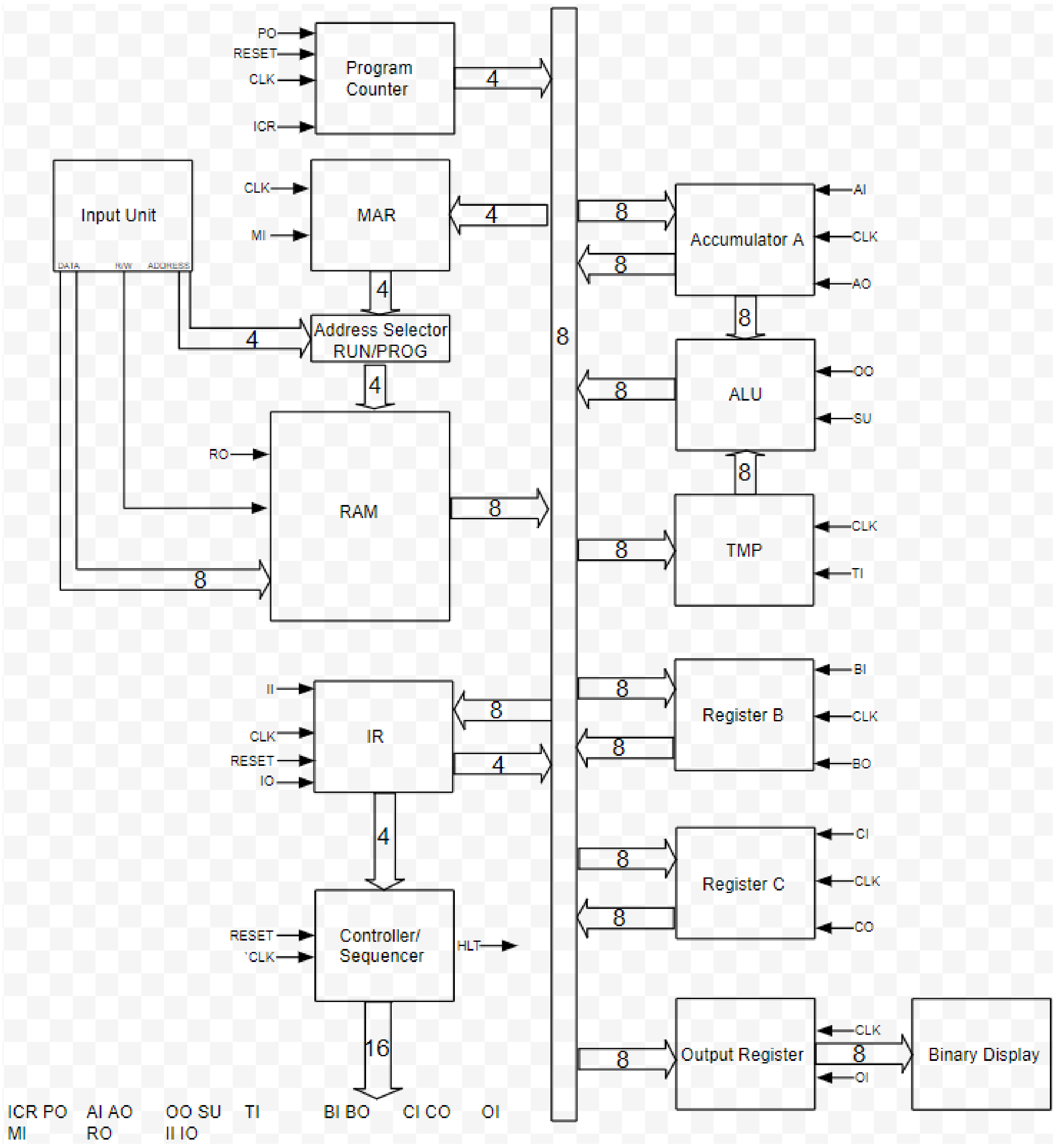
LiteChip-8: A Simple and Educational Microprocessor

Schematic Overview:



Hardware Overview:

- 8-bit Bus
- 16x8 Bit Random Access Memory (RAM)
- 8-bit Arithmetic and Logic Unit (ALU)
- 4-bit Program Counter (PC)
- 4-bit Memory Address Register (MAR)
- Address Selector
- 8-bit Accumulator A
- 8-bit Register TMP
- 8-bit Register B
- 8-bit Register C
- 8-bit Output Register (Output Port)
- Controller-Sequencer: Ring Counter and Control Matrix
- Output Unit: Binary LED display
- Input Unit: Control Board w/ switches
- Clock: 1000 mHz clock signal 50% duty cycle5

Other Important Documents:

This PDF is the overview and documentation of the hardware. The entirety of the documentation is split up between this document, LiteChip-8 Instruction Set, and LiteChip-8 Hardware Specifications.

Documentation:

Program Counter (PC):

We are using four 74LS107 (dual JK flip-flops w/ clear). We could use only 2 for the program counter, but the wiring would be messy and using four allows for expansion and reduces propagation delay.
Wiring:

- CLK: The first chip has the clock input going into its CLK_1 pin. The rest have Q_1 from the last chip, cascading into their CLK_1 pin.
- J_1 and K_1 inputs are shorted together and have one input signal from ICR.
- Q_1 cascade into the next chip's CLK_1 pin, however they also output as Bit 1, 2, 3, 4.
- `CLR_1 pins have `CLR signals going into them.

Tri-state Buffer:

- The outputted binary nibble from the program counter goes into their corresponding A pins on the 74LS126 (a quad bus buffer tri-state chip). Then the corresponding Y outputs go into the bus (I choose the left side of the bus). A high G input spits out the Y outputs onto the bus. Each G input is shorted together and connected to input PO.

Memory Address Register (MAR):

We are using one 74LS173 (4-bit D-type Buffer Register w/ tri-states). The MAR is quite simple, basically just holding data from the program counter to input in the RAM so the RAM knows what memory location to output.
Wiring:

- Data inputs (D1, D2, D3, D4) take in binary nibble from the system bus.
- Data outputs (Q1, Q2, Q3, Q3) spit out binary nibble to address selector.
- CLR is grounded because we do not need to worry about clearing the MAR.
- M and N are the tri-state output control for the MAR, however we want a two-state connection with the MAR and Address Selector (meaning the MAR is always outputting if it has data) so we grounded them since they are active-low to keep data always outputting.
- CLK is obviously connected to the clock.
- `G1 and `G2 are to control the tri-state input of the MAR. Both are shorted together and connected to a NOT gate that receives the MI CON bit. Because the MI bit is active high, but the tri-state input control pins are active low, we needed to add in an inverter. When MI input comes in, the MAR takes in data. Hence a tri-state input for the MAR.

Address Selector:

For the address selector, we are using a 74LS158 (quad 2-to-1 multiplexer). This is necessary so we can switch modes when we are programming or running the computer. It takes in address location nibble from the MAR for run mode, or it takes in data from the input unit, which the operator types out a address location nibble.
Wiring:

- Data input lines A (A1, A2, A3, A4) take in the binary nibble from the MAR.
- Data input lines B (B1, B2, B3, B3) take in the binary nibble from the input unit.
- Data output lines Y (Y1, Y2, Y3, Y4) spit out either A nibble or B nibble to the RAM.
- `G, which controls data storing and is active low, is grounded because we want the address selector always taking in data.

Random Access Memory (RAM):

We are going to use 74189 ICs. These are fully decoded 64 bit non volatile RAM chips. These are structured as 16 x 4 bit chips. Because their outputs are bubbled (inverted) we are using two 74LS04s (hex inverters) to un-invert the outputs. The RAM holds data temporarily to be accessed later by the CPU, however in our case, the RAM will be programmed by hand by the operator. This will hold the instructions and data for the program the CPU will run.

If you may recall, an instruction or line of data for our microprocessor is 8 bits or 1 byte. We are going to split up our RAM into 2 16x4 bit sections. If our byte of data is an instruction, then the upper chip will contain the op code data and the lower chip will contain the address memory

location (if it is a memory referenced instruction). If our byte of data is just data, then the upper chip will contain the upper nibble and the lower chip will contain the lower nibble. For both chips, their address buses will be connected to the address selector output. Their data buses will be connected to the operator's input unit.
Wiring:

- The 8 data lines coming from the input unit are split in half, with the upper lines (D7-D4) going into the upper chip pins (D1-D4); same with the lower chip with D3-D0 going into lower chip pins (D1-D4).
- Prog/Run input line goes into ME (Memory Enable) and Read/Write line goes into WE (Write Enable).
- A1 - A4 pins on both chips receive the same input from the address selector, therefore making the RAM 16 x 8.
- S1 - S4 on both chips make up the 8 bits of output data from the RAM, these go into the hex inverters since the S outputs are complemented (inverted), that way we get proper data coming onto the system bus.

Operation Notes:

- If the operator wants to program the RAM, ME should be switched onto ground, and WE should be switched onto ground. This makes the RAM enabled, and on write mode, ready to intake data from the operator.
- When the computer is running, WE should be switched off of the ground (making it float), which will give it a high input (meaning it is on read mode). ME will be switched off from ground and it will stay high (hold operation) and only will come back to low when RO bit comes through the inverter, making it go low (which enabled memory again), and since it is on read mode since WE is high, it outputs its data onto the system bus.

Instruction Register (IR):

For the instruction register, we are going to use the same chip we used for the MAR, the 74LS173 (4-bit D-type Buffer Register w/ tri-states). We will use two of these chips, one to hold the upper nibble which is the opcode, and one to hold the lower nibble which (if MRI) is the memory address location. The upper nibble opcode gets sent off to the control unit, into the control matrix. The lower nibble memory location gets sent to the bus.
Wiring:

- Data inputs (D1, D2, D3, D4) take in binary nibble from the system bus.
- Data outputs (Q1, Q2, Q3, Q3) spit out binary nibble to address selector.
- CLR is grounded because we do not need to worry about clearing the MAR.
- M and N are the tri-state output control for the MAR, however we want a two-state connection with the MAR and Address Selector (meaning the MAR is always outputting if it has data) so we grounded them since they are active-low to keep data always outputting.
- CLK is obviously connected to the clock.
- `G1 and `G2 are to control the tri-state input of the MAR. Both are shorted together and connected to a NOT gate that receives the MI CON bit. Because the MI bit is active high, but the tri-state input control pins are active low, we needed to add in an inverter. When MI input comes in, the MAR takes in data. Hence a tri-state input for the MAR.

Accumulator A:

We are using two 74LS173 (4-bit D-type Buffer Register w/ tri-states). This is the same chip as the one used for the MAR. We also need to connect the outputs of the accumulator by 2-state to the adder-subtractor. That is easy enough but we also need to be able to output the contents of the accumulator when the signal bit AO is pulsed, so we are going to use 74LS126s (tri-state buffers).

Wiring:

- Data inputs (D7 - D0) take in binary bytes from the system bus and go into D1 - D4 on both chips.
- Data outputs (D7 - D0) put data onto the bus from the tri-state buffers, running from Y1 - Y4 on both buffer chips.
- Data outputs (D7 - D0) put data into the adder-subtractor running form pins (Q1 - Q4) from the 75LS173s.
- CLRs are grounded since we do not need to clear the accumulator
- M and N are the tri-state output control for the accumulator, however we want a two-state connection with the accumulator and adder-subtractor so we grounded them since they are active-low to keep data always outputting.
- CLK is obviously connected to the clock.
- `G1 and `G2 are to control the tri-state input of the MAR. Both are shorted together and connected to a NOT gate that receives the AI CON bit. Because the AI bit is active high, but the tri-state input control pins are active low, we needed to add in an inverter. When AI input comes in, the accumulator takes in data. Hence a tri-state input for the accumulator.

Adder-Subtractor:

We are going to use 74LS83s (4-bit binary full adders). We could just use some XOR, AND, and OR gates, however that will ruin the compactness and simplicity of the schematic. To have both the addition aspect and the subtraction aspect of our ALU, we need to add XOR gates to the B input on the adders, and have the SU CON bit connected to the 2nd input on those XOR gates and the carry in on the first adder chip. Therefore, with the SU bit off, the ALU performs addition as it should. When the SU bit is on, the B inputs to the adders are inverted and subtraction is performed.
Wiring:

- Input data lines (A0 - A7) go into the A pins with each 74LS83 chip.
- Input data lines (B0 - B7) run through XOR gates into B pins with each 74LS83 chip.
- The first nibble chip (Bit 1 - Bit 4) should have its C0 pin connected to SU CON bit. Then its C4 pin should run to the C0 pin on the second nibble chip.
- The second nibble chip (Bit 5 - Bit 8) should have its C4 pin disconnected
- Both chips Σ outputs should be connected to the tri-state buffers (buffers output lines Bit 1 through Bit 8 should go to bus).
- XOR gates should invert B inputs if SU is enabled.

Register B and C:

Both general registers that are used in this microprocessor are using 74LS173 chips, and are nearly identical to the TMP Register. However both data and address buses are hooked up to the system bus, and input and output control bits are used. These registers are used for bytes to be stored and used later on for computations.
Wiring:

- Input data lines (D1 - D8) go into the D pins with each 74LS173 chip.
- Output data lines (Q1 - Q8) come from the Q pins with each 74LS173 chip.
- CO and BO con bits go through inverters to the M and N pins with each 74LS173 chip.
- CI and BI con bits go through inverters to the `G2 and `G1 pins with each 74LS173 chip.
- CLK goes to the clock pin.
- CLR is grounded as we do not need to worry about clearing the general registers

Output Port and Binary Display:

The output port, also known as the output register, is identical to the TMP Register. The outport port takes data when its OI con bit is active and runs it to the binary display. The binary display is made up of 8 light emitting diodes. This will show the operator the results of whatever computations his program finished.
Wiring:

- Input data lines (D1 - D8) go into the D pins with each 74LS173 chip.
- Output data lines (Q1 - Q8) come from the Q pins with each 74LS173 chip and go into the binary display.
- OI con bit goes through inverters to the M and N pins with each 74LS173 chip.
- The `G2 and `G1 pins with each 74LS173 chip are grounded to keep the output 2-state.
- CLK goes to the clock pin.
- CLR is grounded as we do not need to worry about clearing the output port.

Clock Signal:

Chip C28 is a 555 timer. The way we wired it creates a 2 kilohertz 75% duty cycle. The flip-flop turns that into a 1 kilohertz or 1000 hertz 50% duty cycle. The output is buffered through 2 inverters.
Wiring:

- 36k ohm and 18k ohm are together in a series, with a 0.01 uF non-polarized capacitor at the end, connecting back to ground.
- Pin 7 connected between the 36k and 18k resistors.
- Pin 2 and 6 connect between the 18k resistor and 0.01 uF capacitor.
- Pin 5 connected to ground through 0.01 uF capacitor.
- Pin 3 connected to the clock of the 74LS107 flip-flop.

- The flip-flop J and K inputs are connected together, taking input through inverted con bit HLT.
- Q is the output of the clock signal. Goes through 2 inverters (buffer) to the signal line.

- `CLR has an inverted clear signal going through it.

Control Unit:

The control unit consists of a 6-bit ring counter and a control matrix. The ring counter (using 74LS107 chips; same as program counter) acts to be an overarching controller of what bits can activate at what T states. We have 6 T states, therefore we need a 6-bit ring counter. The control matrix decodes the binary nibble coming in from the instruction register and activates the bits that the opcode has.

Control Matrix:

The lines our AND gates will be connecting to has the binary opcode nibble and the binary opcode complement nibble. If we want a specific AND gate to activate based on a specific binary nibble, we need to arrange each and every AND gate in that manner. Lets take for example the instruction ADD B. The opcode for this instruction is 0001. So 0001 comes through on lines D7, D6, D5, and D4. AND gates only activate if all inputs are 1. So on the AND gate, input 4, 3, and 2 need to be set on the complement of D7, D6, and D5. And for input 1, it should be on D4. Therefore, when 0001 comes through, this specific AND gate will activate.

AND gates are labeled by the macro-instruction that is being run and the T state. For example, the LDA T5 AND gate has one input connected to the LDA line, and the other input connected to the T5 line. Each of these AND gates will be connected to an OR gate that is labeled with a micro-instruction, and depending on how many of those specific microinstructions are throughout the instruction set, that will determine how many AND gates are connected to their OR gate. For example, the OR gate labeled with microinstruction RO has LDA T5, MVI B T4, Memory State T3, and MVI C T4 connected to its 4 inputs. (Since memory state T3 isn't specific to any macro-instruction, it simply is just connected by a line to the OR gate).

Ring Counter:

The ring counter as: 000001, 000010, 000100, 001000, 010000, 100000, then resets, which is unlike ripple counters or synchronous counters. This coordinates the T states, making only the micro-instructions specific to their T state runs when their T state is active.
Wiring:
The Q output goes to J, `Q output goes to K, CLK takes in clock signal, and `CLR takes in active low clear signal. However the last chip has its Q and `Q going to the first chip J and K. I won't get into the specific wiring of the control matrix as it would be too much to specify, very redundant, and the actual schematic will be more useful than a text explanation.

Control Unit:

The control panel is the entirety of the input unit. It is used by the operator to clear out system blocks, program data and specific memory locations with the RAM, and operate control signals with the address selector and RAM. Keep in mind, for the data and address switches, closed connects the ground with the line, creating a logical 0. Open creates a floating pin, which

creates logic 1 (this typically occurs in most gates where they will float towards a high state for TTL Bi-polar transistor circuits). Obviously any electrical noise can create interference and cause issues.
Wiring:

- The `A/B switch controls the address selector. Close the switch to have run mode, open the switch to have program mode.
- The A1 through A4 switches specify what memory location you want to input into with the RAM.
- The D7 through D0 switches specify what opcodes, operands, and data you want to program into your specified memory location.
- The Prog/Run switch is to turn the RAM on or off. Because the ME input pin is active-low, you will turn it on by closing the switch and connecting it with ground. This will allow you to start programming. Opening the switch will float the pin, disabling the RAM. This is run mode, where the RAM only turns back on when con bit RO comes in through the inverter.
- Read/Write switch is to toggle between programming and running the chip. Closing the switch will ground the active low pin, and allow you to program it. Opening the switch will put the RAM into run mode. It will float the active-low pin and will turn the RAM on read mode, which means it will output its data onto the bus.
- The clear switches should be used when you boot up the computer. Simply switch each one closed (exception made for IR, which should be switched open to clear). Then toggle them back.
- The HLT switch is your final “run” switch. When the switch is open, a HLT signal is run down to the clock circuit, preventing it from outputting its clock signal. When the operator finished programming the RAM, cleared all the necessary system blocks, and is ready to run their program, simply just close the switch to get rid of the HLT signal and the clock will start up.

Operation:

The first thing you want to do is make sure all the switches on the control board are in their ideal state. Go from the top control switch to the bottom: `A/B should be open, don’t worry about the address switches, PROG/RUN should be closed, Read/Write should be open (this is because until we have our data and address switches ideal for each memory location, we don’t want to write), don’t worry about the data switches, and switch the clear signals once (PC, Ring, CLK should be back open; IR should be closed). Now make sure HLT is open, we don’t want to run the CPU until we have programmed the RAM.

Now to program the RAM: Let's start with the first location of the RAM, 0000 (0H). Set all the address switches closed. Now proceed to program your byte into the memory location, for an example this could be LDA 7H, which would be entered as 0000 0111. This byte would instruct the CPU to load the accumulator with data from memory location 0111 or 7H from the RAM.

Now when you are satisfied with your instruction, switch Read/Write closed. Then switch it back to open. Congratulations! You programmed your first memory location! Now continue to do this with all 16 memory locations, planning out your program to accommodate MRIs, etc.

Conclusion:

This project was created to wrap up my studying of Digital Computer Electronics by Albert Paul Malvino. It was inspired by the SAP-1 microprocessor on Chapter 10. After 120 pages of notes, I decided it would be fit to complete a final project. I intend for this project to be purely educational. It was a very minimal microprocessor (hence the name LiteChip-8) and is not universally computational (or turing complete). There is no user interaction with the microprocessor while it is running, and simply can be programmed, run instructions, and perform calculations. I eventually want to expand my studies and work into developing a universally computational microprocessor. I am almost positive there are some small errors here and there throughout the design, and will need to be troubleshooted if necessary. If you have any suggestions, comments, or error corrections that you would like to inform me of, feel free to reach out!