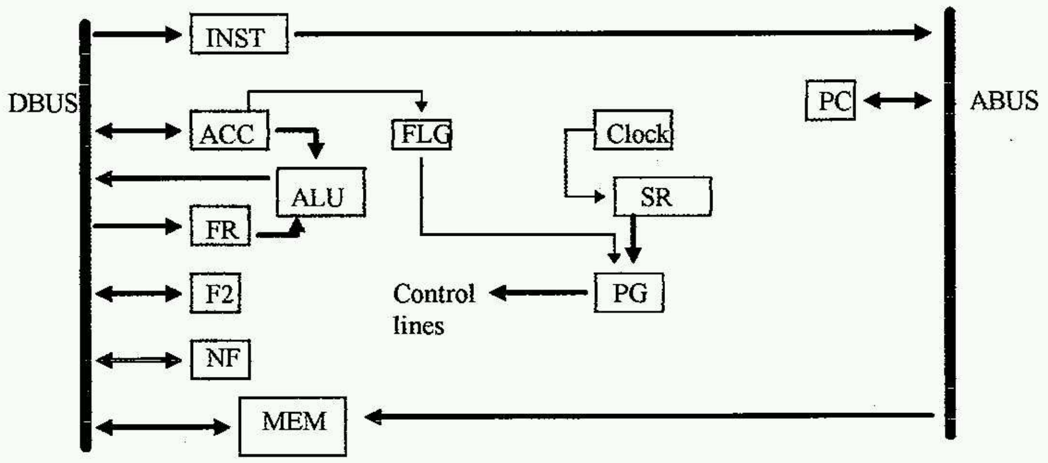
Michael Robinson
Honors Computer Seminar - 2

April 10, 1997
Quarterly Report

Update:

This week, the clock clk and clk$_3$ oscillators have been completed. The problems were two-fold. First, I wired it wrong; and second, the resistors and capacitors need to be correctly "balanced." The "balance" was not forseen, and has been compensated. clk$_3$ runs several times faster than clk, thus allowing store timing functions to work correctly.

Progress to Present:



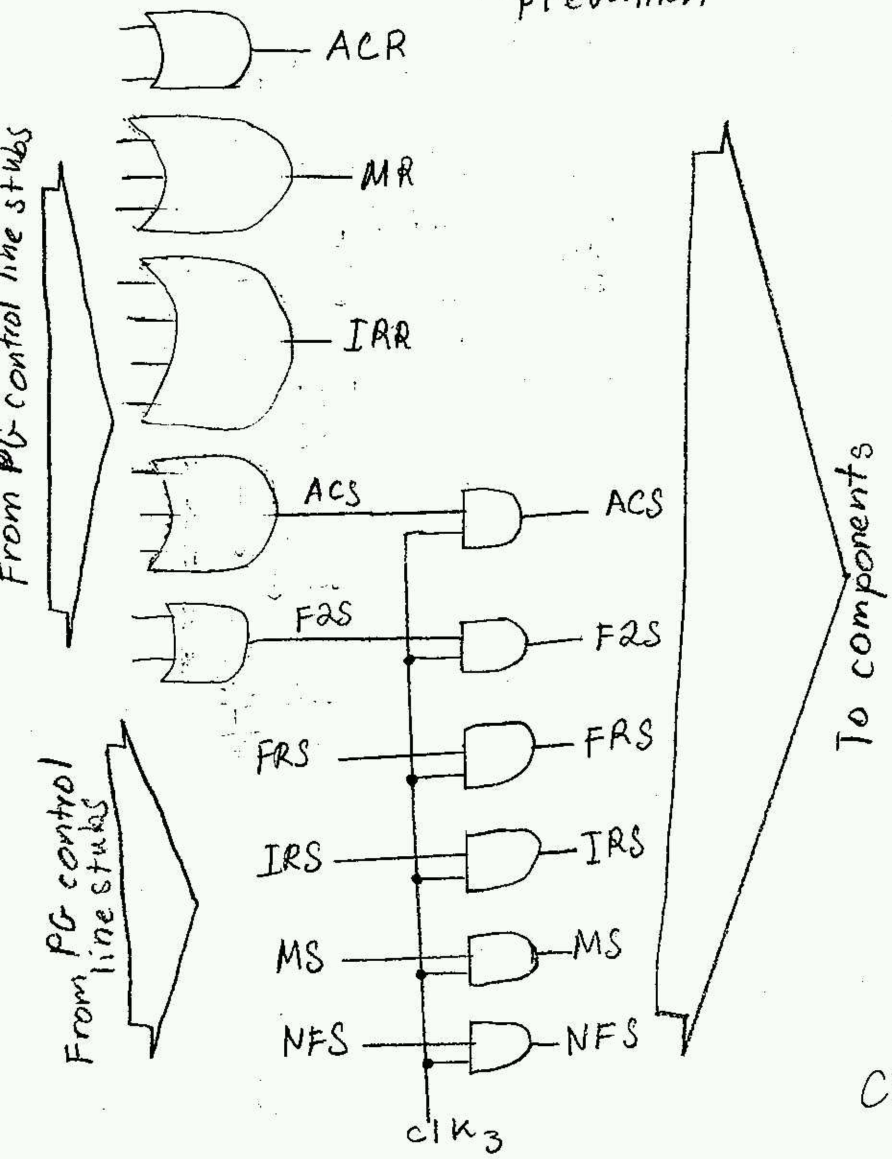
Note: Those items highlighted in red have been completed.

The above chart shows my progress over the past quarter. As you can see, I have completed most of the components in the computer. What remains to be done is the front panel and integration. Integration will include linking busses, control lines, and ultimately, the boards. I am quite pleased with the progress that I have made, considering the difficulty of this project.

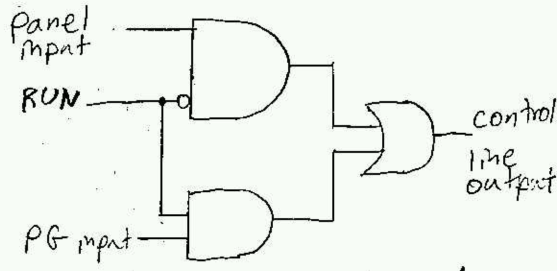
What I have learned (or should have):

- Planning is essential to the completion of a complex project. Although I did not attempt to do this project without plans, I'm quite sure that it wouldn't work. Flexible plans are better than rigid ones, though. Many times, I needed to modify the plans for a variety of reasons. I also am quite sure that planning can't cover every contingency. The more problems fixed ahead of time the better, but I am sure to expect problems (especially when I make the plans!).
- Synchronous chips are not a problem if you wire them as such. Essentially, this applies to anything. "Don't try to force a square peg into a round hole" is a common version. Synchronous chips like more than one clock to time their actions, making an resistor-capacitor delay is not an option. This is the reason why I have clk$_3$.
- Wires combust their insulation when more than 10 amps go through them. Essentially, short circuits with thin wires are good for incense. It is a good thing that the power supply's circuit breaker pops at 20 amps. . .
- Noise annoys. Many timing-intensive and synchronous ciruits get confused with noise. In several instances, this was the source of a problem. The manual switching produces lots of

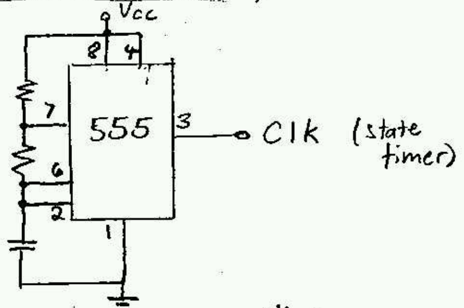
Control line Management - signal timing and interference prevention



Control line interface - connecting panel to internal circuitry

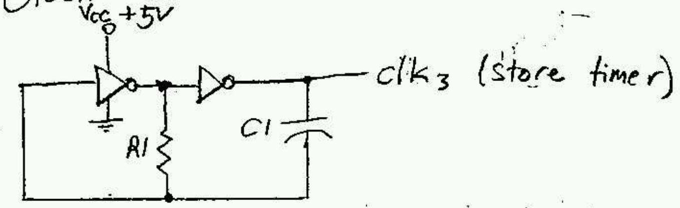


The following panel inputs are used:
PCS, PCR, reset, MS
MR, ACS, ACR

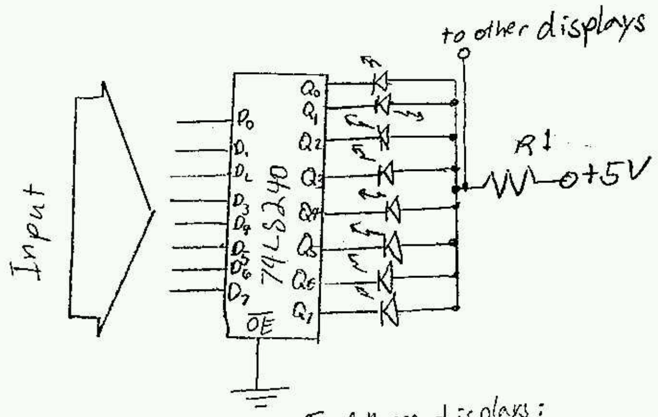


Alternate clock oscillator

Clock oscillator - system synchronization

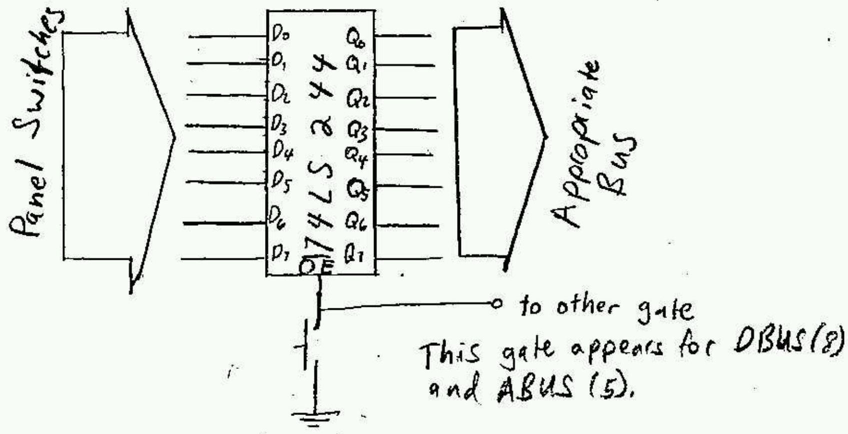


Display - Output to panel



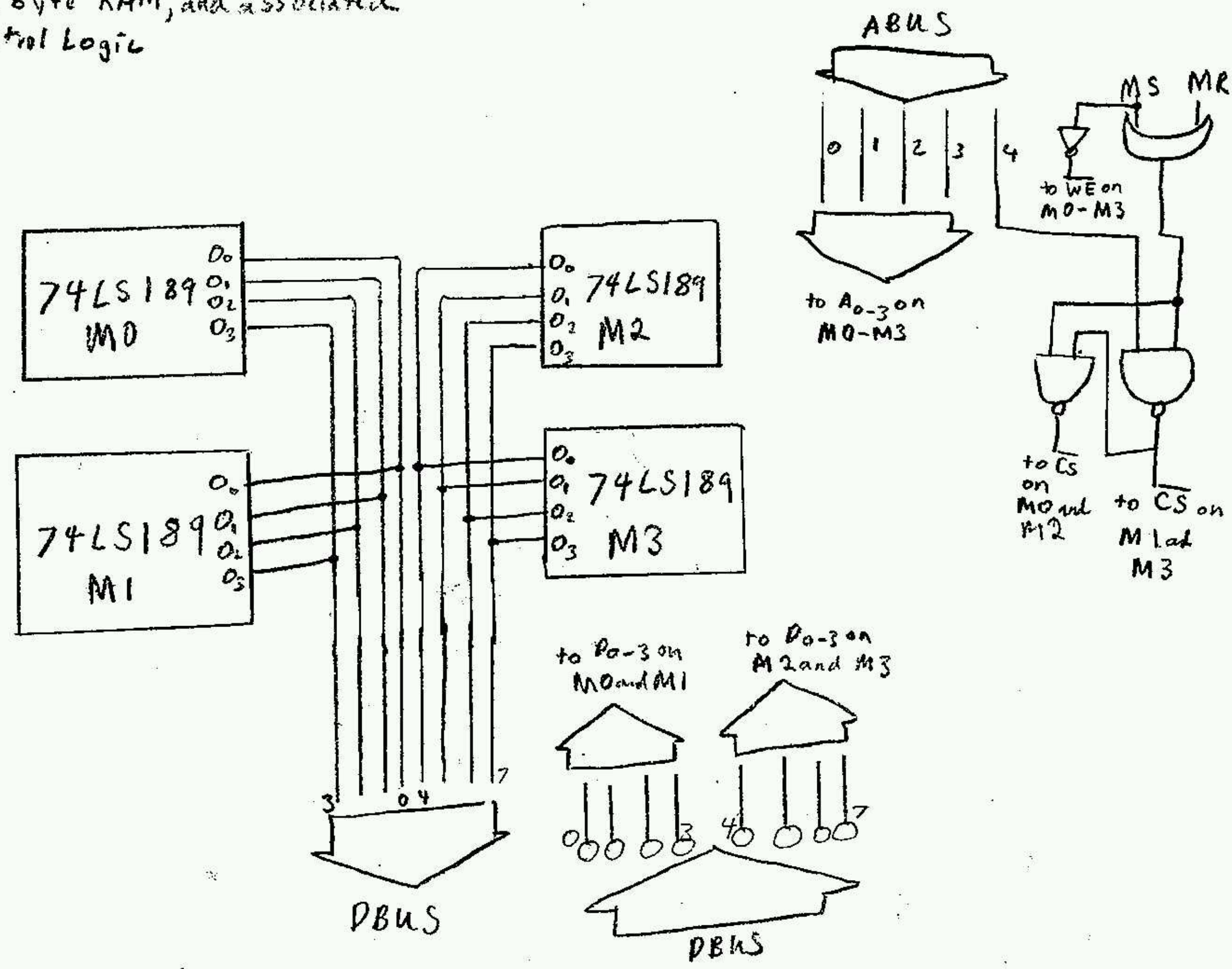
There are 5 of these displays:
Source (number of inputs)
IR (8) ABUS (5)
DBUS (8) PC (5)
SR (6)
R1 value: ~ 5Ω

Front Panel Input

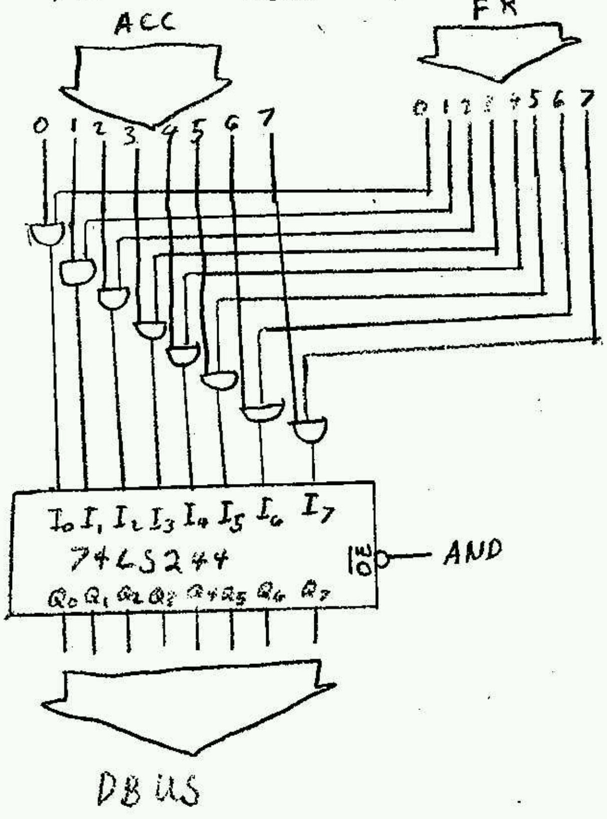


This gate appears for DBUS (8) and ABUS (5).

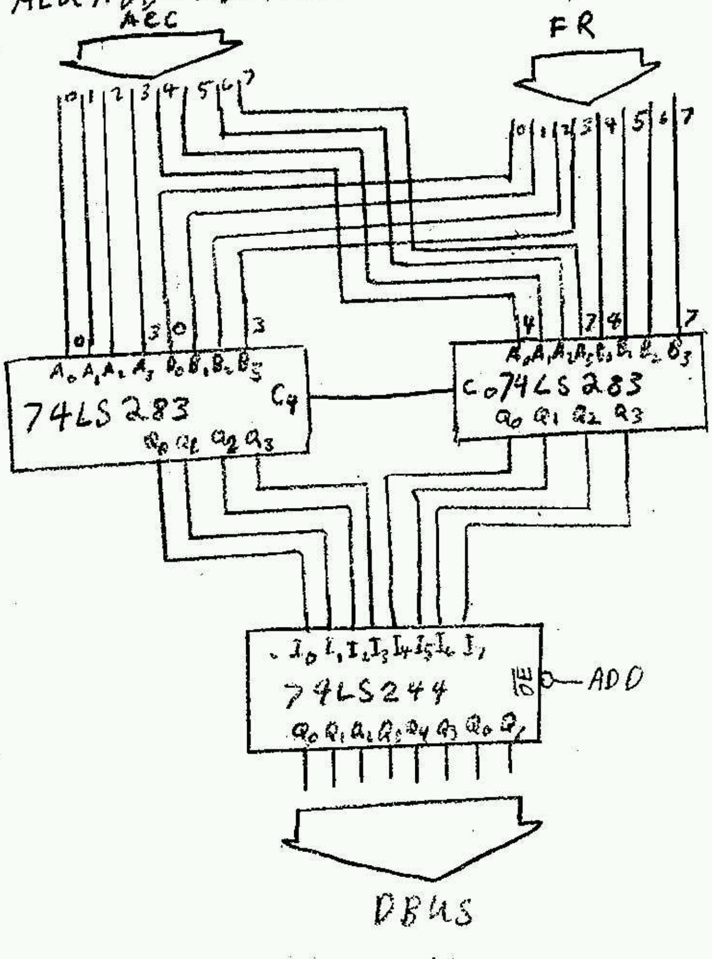
MEM- 32 byte RAM, and associated control Logic



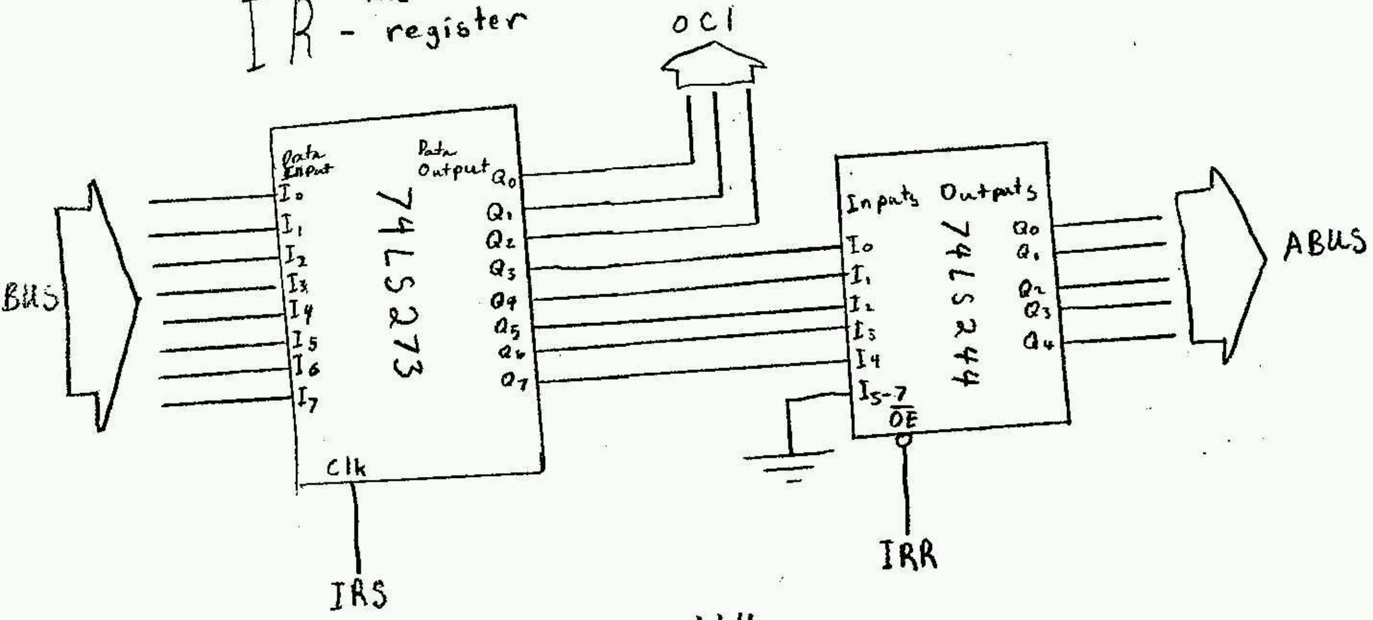
ALUAND- AND Portion of the ALU


ACC
FR
0 1 2 3 4 5 6 7
0 1 2 3 4 5 6 7
I_0 I_1 I_2 I_3 I_4 I_5 I_6 I_7
74LS244
$\overline{OE}$
AND
Q_0 Q_1 Q_2 Q_3 Q_4 Q_5 Q_6 Q_7
DBUS

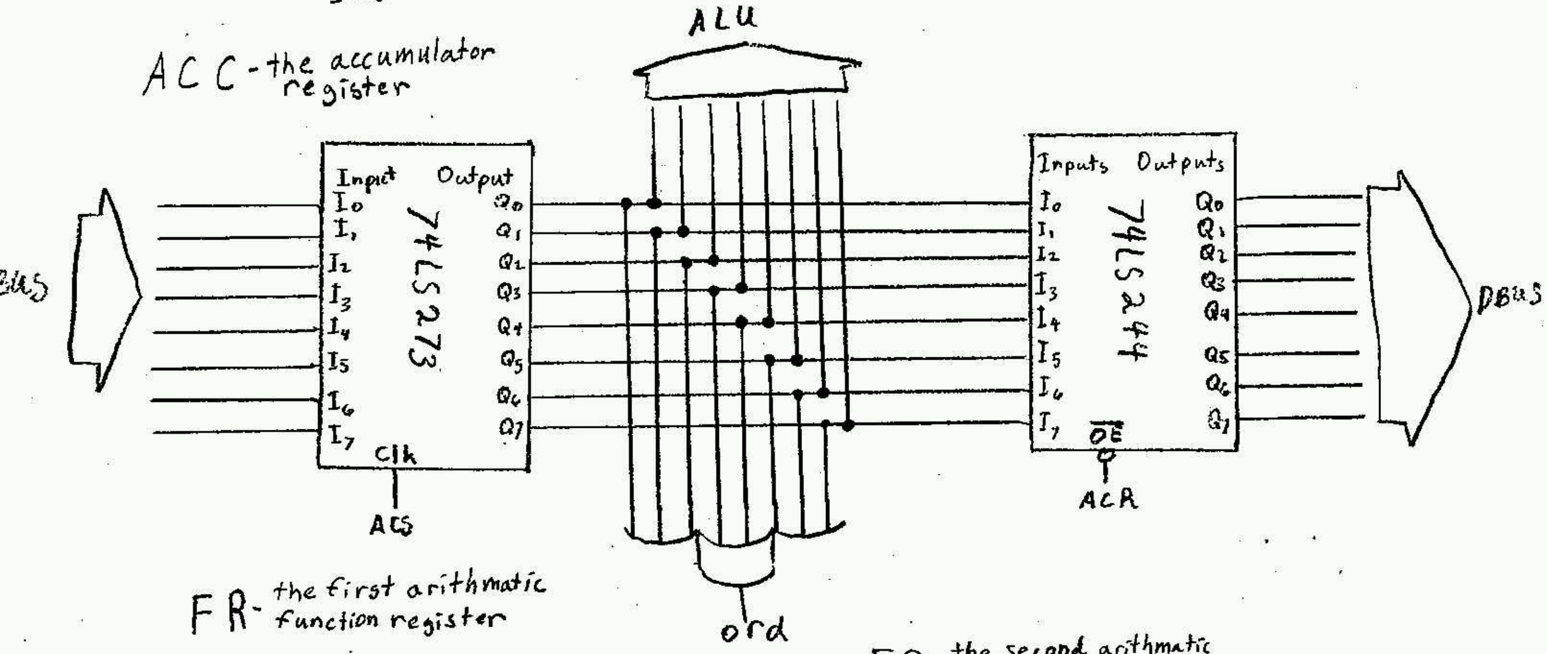
ALUADD - ADD Portion of the ALU


ACC
FR
0 1 2 3 4 5 6 7
0 1 2 3 4 5 6 7
0 3 0 3
4 7 4 7
A_0 A_1 A_2 A_3 B_0 B_1 B_2 B_3
74LS283
C_4
C_0 74LS283
Q_0 Q_1 Q_2 Q_3
I_0 I_1 I_2 I_3 I_4 I_5 I_6 I_7
74LS244
$\overline{OE}$
ADD
Q_0 Q_1 Q_2 Q_3 Q_4 Q_5 Q_6 Q_7
DBUS

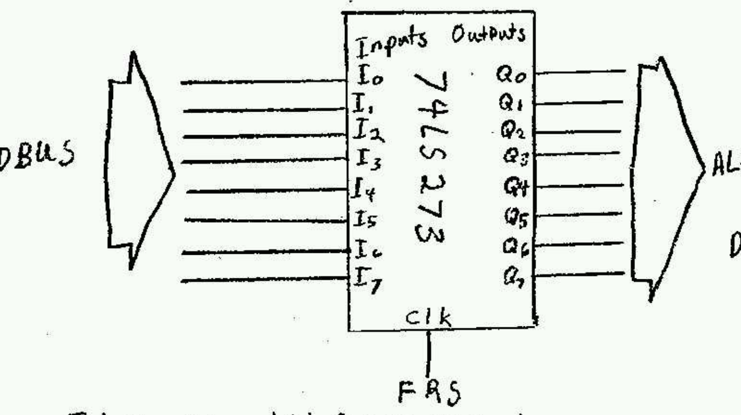
IR - the instruction register



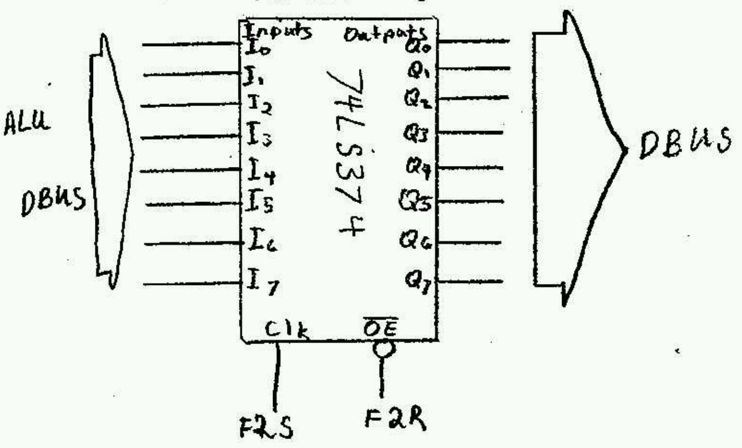
ACC - the accumulator register



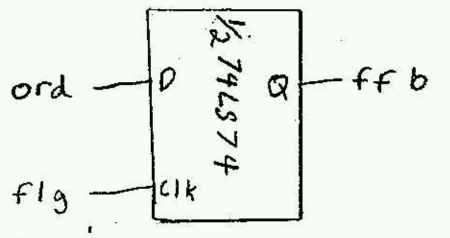
FR - the first arithmatic function register



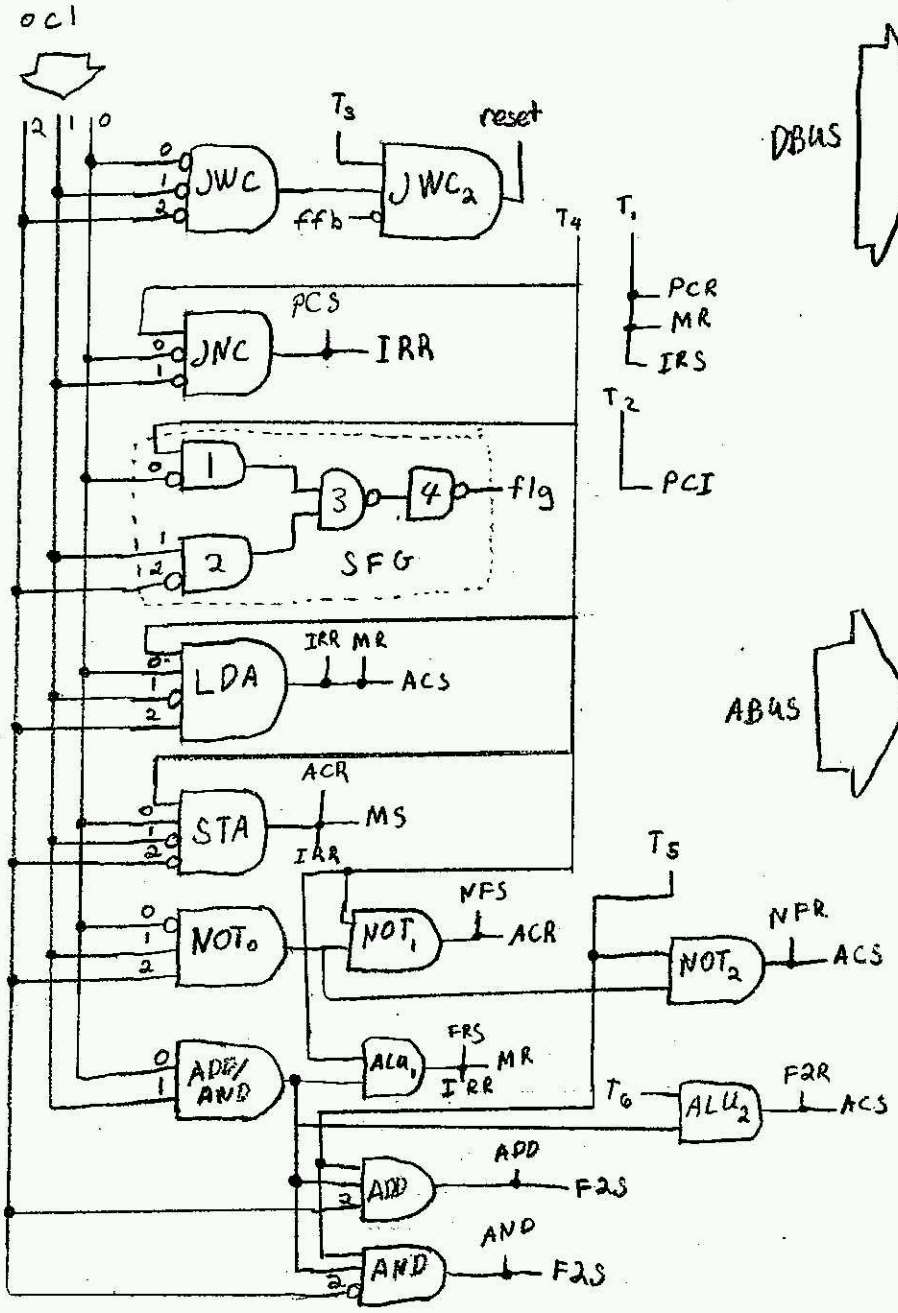
F2 - the second arithmatic function register



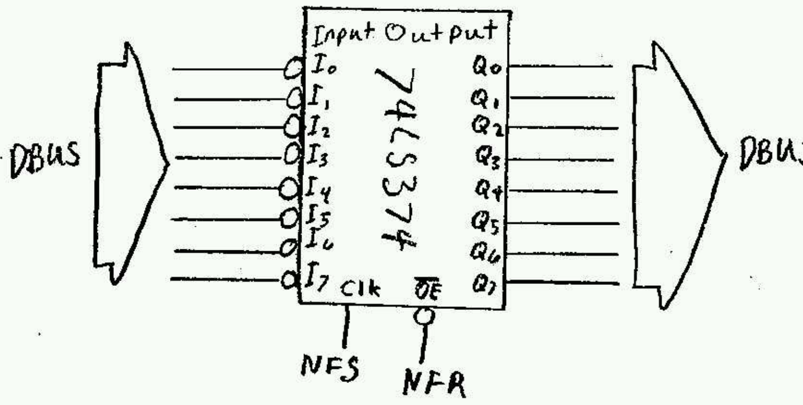
FLG - Flag bit for conditional Jumps



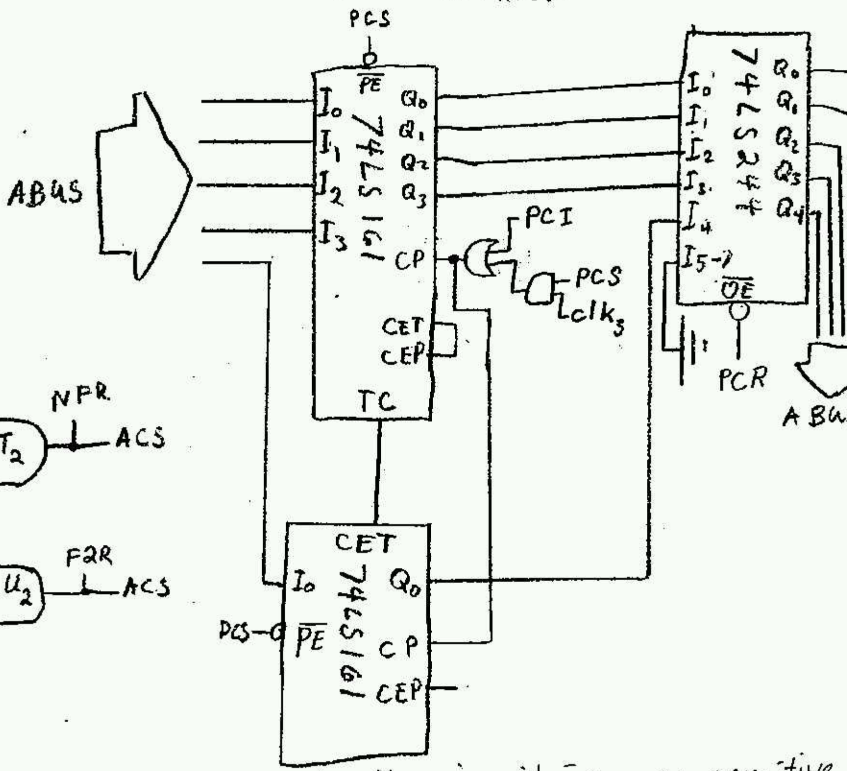
PG - controller of control lines and instruction processing



NF - NOT arithmetic function register

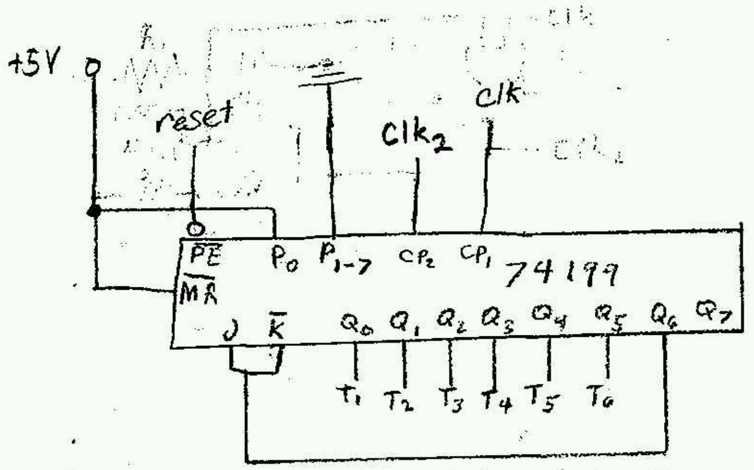


PC - the program counter



note: This circuit is very sensitive to noise on control lines PCS, PCI, and the clock clk3

SR - State Register



Specifications of Computer Design:

Components:

ACC - The accumulator register - This register is used for memory transactions, logic, and conditional instructions. It is the most used register, and is general purpose.

IR - The instruction register - This register holds the current instruction. The register is broken in two pieces. The first piece is 3 bits, and holds the opcode of the instruction. The remaining 5 bits are used for the argument for the instruction. The opcode is send to the PG via the lines titled "ocl". The argument is attached to the ABUS with a bus driver (74LS244).

ABUS - The address bus - This 5 bit bus serves the purpose of providing an address for the memory, and for transactions between the IR and PC.

DBUS - The data bus - This 8 bit bus connects the ACC, FR, F2, NF, IR, and MEM. It allows transactions between these components.

FR, F2, NF - Arithmetic function registers - These registers are special purpose for logic and mathematical calculations. FR is connected directly to the ALU, for adding and anding. Both F2 and NF are connected to the DBUS, and serve as temporary storage during the execution of arithmetic functions. NF inverts its input.

ALU(ADD,AND) - Arithmetic and Logic Unit - This component consists of two major sections, the ADD and AND sections. The ADD section contains two 4 bit adders that add the contents of ACC and FR. AND ands FR and ACC. Both outputs are gated to the DBUS.

PG - Control Line Pulse Generator - This device contains logic that controls the operation of the machine. Logic controls the triggering of the control lines at each of the six machine states. Note that the SFG portion is consisted of several 2 input NAND and AND gates, rather than a single 4 input AND. The reason is that I did not buy an extra chip of 4 input ANDs. This would leave one gate unused, and I have extra NANDs and ANDs that will suffice.

SR - State Register and power-up circuitry - This block of components controls the state of the machine. There are six states: the first two for loading instructions, the third for conditional jumps, and the remaining three for normal instructions. The power-up circuitry serves to clear the shift register that determines the state. There is also some logic to control the clock, which signals state changes.

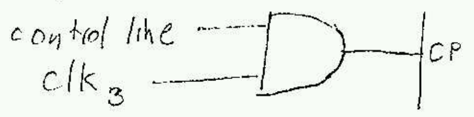
FLG - Flag Bit - This flip-flop is used to control the operation of conditional jumps. If set, a conditional jump will jump. If cleared, the jump will be ignored. It is set by the SFG instruction, which sets this flip-flop if any bit in the ACC is nonzero.

PC - Program Counter - The program counter points the current instruction in memory. It consists of two 4-bit counters linked together. The last three bits of the second counter are ignored. This makes the PC 5 bits long in size, and enables it to be connected to the ABUS. It can be incremented or set directly.

MEM - 32-byte Memory - This is the main memory in the machine. It is broken into thirty-two 8 bit pieces, each with a unique address. It is connected both to the DBUS, and ABUS. When asked to perform a memory function, it reads the address off the ABUS, and uses the DBUS for I/O. There are 4 memory chips, each holding 64 bits.

Block Diagram:

Store control lines:



Control Lines:

There are many control lines, which control the functions of each component. I have a convention for naming control lines by their function. The first two characters of the control line's name is the first two characters of the component that it controls. The next character is R, which means "make your contents available on the bus" or S, which means "take the contents of the bus, and store it." There are several control lines that do not follow this convention, but most do.

PCR - Read contents of the PC onto ABUS
PCS - Store the contents of ABUS in PC
PCI - Increment the contents of PC

ACR - Read the contents of ACC onto DBUS
ACS - Store the contents of DBUS into ACC

IRR - Read the argument part of IR onto ABUS
IRS - Store the contents of DBUS into IR
ocl - brings the opcode of the current instruction to PG

NFR - Read the contents of NF onto DBUS
NFS - Store the contents of DBUS into NF

FRS - Store the contents of DBUS into FR

F2R - Read the contents of F2 onto DBUS
F2S - Store the contents of DBUS into F2

flg - set FLG if any bit in the ACC is nonzero
ord - is set if any bit in ACC is nonzero
ffb - is set if FLG is set

AND - Put the anded result of ACC and FR on DBUS
ADD - Put the sum of ACC and FR on DBUS

MS - Store the contents of DBUS into MEM at address in ABUS
MR - Read the contents of MEM onto DBUS at address in ABUS
clk_3 - a faster tick than clk, signals [illegible] SR [illegible] clk (and all store control lines)
clk - the clock tick, after passing through a control panel switch, signals state change
clk_2 - the single step clock tick, actually a switch connected to ground
reset - reset the SR to state T_1
T_1 - is set if machine is in first state
T_2 - is set if machine is in second state
T_3 - is set if machine is in third state
T_4 - is set if machine is in fourth state
T_5 - is set if machine is in fifth state

T_6 - is set if machine is in sixth state

<u>Instructions:</u>

Listed by Mnemonic:

JWC - Jump to the address specified in argument portion of instruction if FLG is set. Go to next instruction if FLG is cleared.
JNC - Unconditional jump to the address specified in the argument portion of the instruction.
SFG - Set FLG if any bit in the ACC is nonzero, else clear FLG. No arguments are used.
LDA - Load the ACC with the contents of MEM at the address in the argument.
STA - Store the contents of ACC in MEM at the address in the argument.
NOT - Bitwise NOT every bit of the ACC, store the result in the ACC.
ADD - Add the contents of the ACC to the contents of MEM at address in argument, store result in ACC.
AND - Bitwise AND the contents of the ACC to the contents of MEM at address in argument, store result in ACC.

Listed by Opcode:

The sequence of control lines is a representation of how the PG will set control lines. Control lines set on the same state are separated by commas. Semicolons separate states. Most instructions start at the fourth state. However, JWC starts at the third state. If FLG is not set, the processor is forced into the first state. In the chart below, the third state is represented by "(conditional)". The Load Prefix occurs on the first and second states of every instruction. It loads the instruction into IR.

Machine Code	Mnemonic	Sequence of Control Lines
000	JWC	(conditional);PCS,IRR
001	JNC	PCS,IRR
010	SFG	flg
011	NOT	ACR,NFS;ACS,NFR
100	STA	MS,ACR,IRR
101	LDA	MR,ACS,IRR
110	AND	FRS,MR,IRR;AND,F2S;F2R,ACS
111	ADD	FRS,MR,IRR;ADD,F2S;F2R,ACS
Load Prefix		PCR,MR,IRS;PCI

<u>Display/Control:</u>

The display/control panel will be a set of LEDs and switches to control the operation of the computer. There are

LEDs for each of the following ABUS, DBUS, PC, IR, and each of the six states. Switches will allow writing to the DBUS or ABUS, single stepping, running, and the power switch. I intend to have a dial to control the run speed (less than 10 kHz). There will be several push buttons connected directly to control lines. Using these buttons, information can be moved to the MEM, ACC, and PC.