CPU Simulation Project

CSc 142: Advanced Computer Architecture, Dr. Faroughi
School of Engineering and Computer Science
California State University, Sacramento

Students: Kim Berry, Miwa Nakamura

The American Society for Engineering Education

Pacific Southwest Section

1989-90 Undergraduate Design Engineering Student Contest

Design Project

Table of Contents

I. Introduction

This project is a fully functional simulation of a Central Processing Unit (CPU) running on logic simulation software called *Logicworks*.[1] *Logicworks* runs on a *Macintosh IIx*. A functional CPU could be constructed from basic logic devices by following this project's design.

This CPU is designed exclusively from standard gates, TTL ICs, a RAM, and a ROM. During simulation, the CPU executes an instruction in about three seconds, and *Logicworks* permits the CPU clock to be single-stepped, allowing scrutinization of each internal control signal, micro-instruction, and data transfer.

The CPU features an 8-bit address bus, 16-bit data bus, nine macro (assembly) instructions, and four internal registers. Data is output by storing to address 80H (memory-mapped I/O method).

Manual DMA is used to modify the RAM. By using the mouse to flip switches in the *Logicworks* simulation, the user enters values/instructions into RAM as follows.

1. Place the CPU in RESET mode (RESET = 1).
2. Enable RAM address bypass (BYPASS = 1).
3. Select address at hex keypads.
4. Select data at hex keypads.
5. Set RAM to write mode (R'/W = 0).
6. Enable data keypad tri-state (OE' = 0).
7. Repeat steps 3-6 as necessary.
8. Disable RAM address bypass (BYPASS = 0).

The program can then be executed by placing the CPU in RUN mode (RESET = 0).

Micro-programmed ROM code for each macro-instruction, and macro-code (machine/assembly language) for a bubble-sort program is included in this document.

[1] *Logicworks* is written by Capilano Computing (501-1168 Hamilton St., Vancouver B.C. Canada, (604) 669-6343)

II. Definitions

AC	Accumulator: Intermediate Register for Compare, INC, or DEC.
C(n)	Control lines in micro-code. Enables hardware actions.
CS	Control Select, controls how MPC determines next address.
IR	Instruction Register: contains macro-code instruction.
macro-code	machine code / user-provided assembly language.
MADR	Address line -> Memory Register.
MDR	Memory -> Data Register.
micro-code	internal CPU instructions
MIR	Micro Instruction Register: contains CS, NA, S, C.
MPC	Program Counter for micro-code.
NA	Next Address in micro-code.
OPND	Operand: address or data value in IR.
PC	Program Counter for macro-code.
R-MUX	Determines whether R1 or R2 will be used to select Register.
R1	Register 1: 00 = A, 01 = B, 10 = C, 11 = D.
R2	Register 2: 00 = A, 01 = B, 10 = C, 11 = D.
S(n)	Select lines in micro-code. Controls MUX's and tri-states.

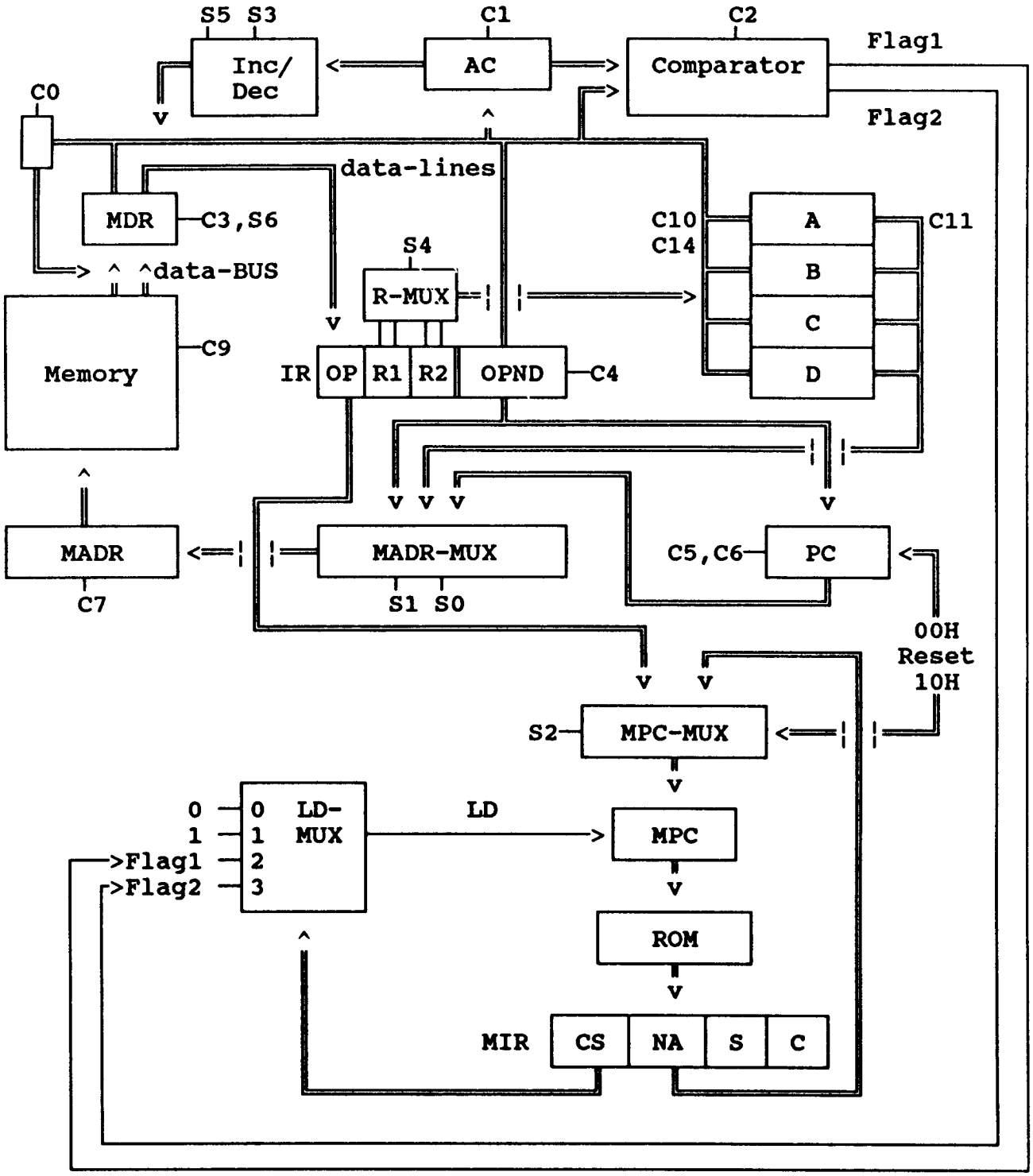
III. IC Descriptions

The following is a description of the ICs used in this CPU simulation. All of these ICs are standard TTL logic devices.

74LS85	4-Bit Magnitude Comparator.
74LS139	Dual 1-of-4 Decoder/Demultiplexer.
74LS153	Dual 4-Line to 1-Line Multiplexer.
74LS157	Quad 2-input Data Selector/Multiplexer.
74LS193	Presettable 4-Bit Binary Up/Down Counter.
74LS244	Octal Buffer (3-state).
74LS374	Octal D Flip-Flop (3-state).

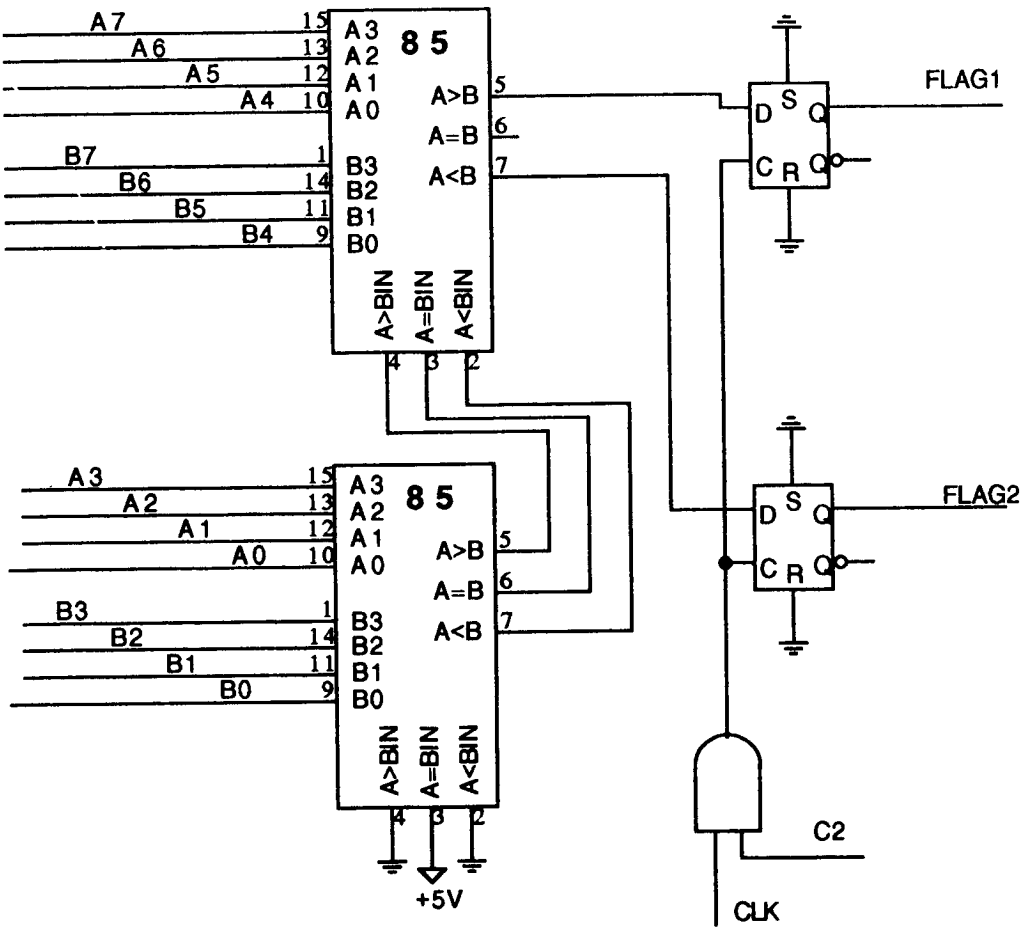
In addition to the above ICs, the CPU utilizes various generic gates and flip-flops, as well as a RAM and a ROM, in its design.

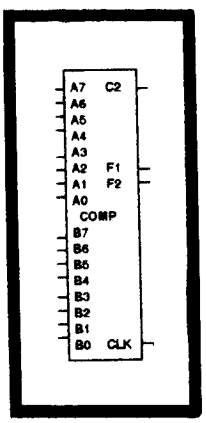
IV. CPU Overview Diagram



V. Comparator -- Detailed Diagram

The comparator is constructed by linking two 74LS85s to form an 8-bit digital comparator. When a clock pulse occurs and control line C2 (Compare AC/Reg) is high, the result on the 74LS85 will be clocked into the D flip-flops which contain FLAG1 (AC > Reg) and FLAG2 (AC < Reg).



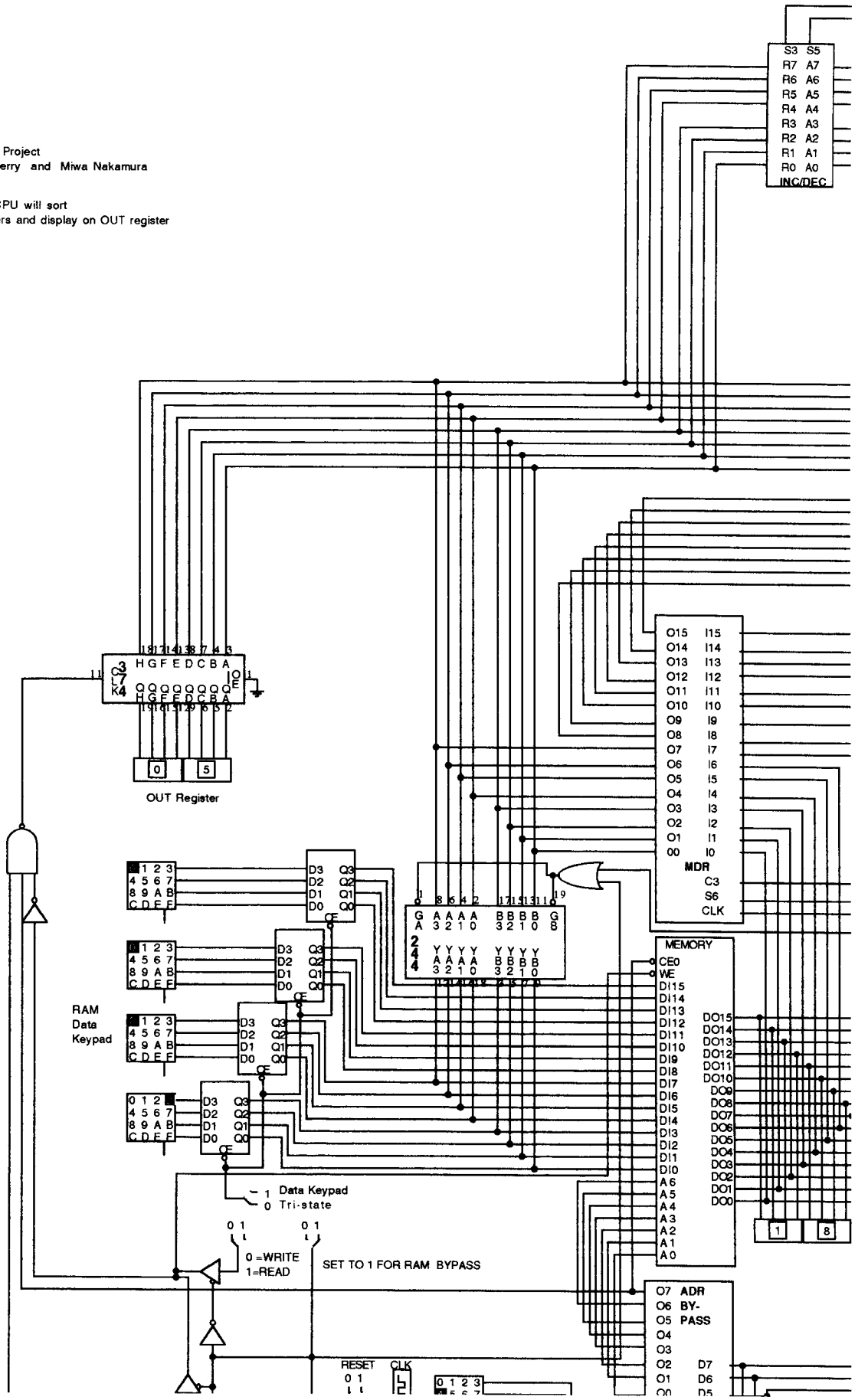


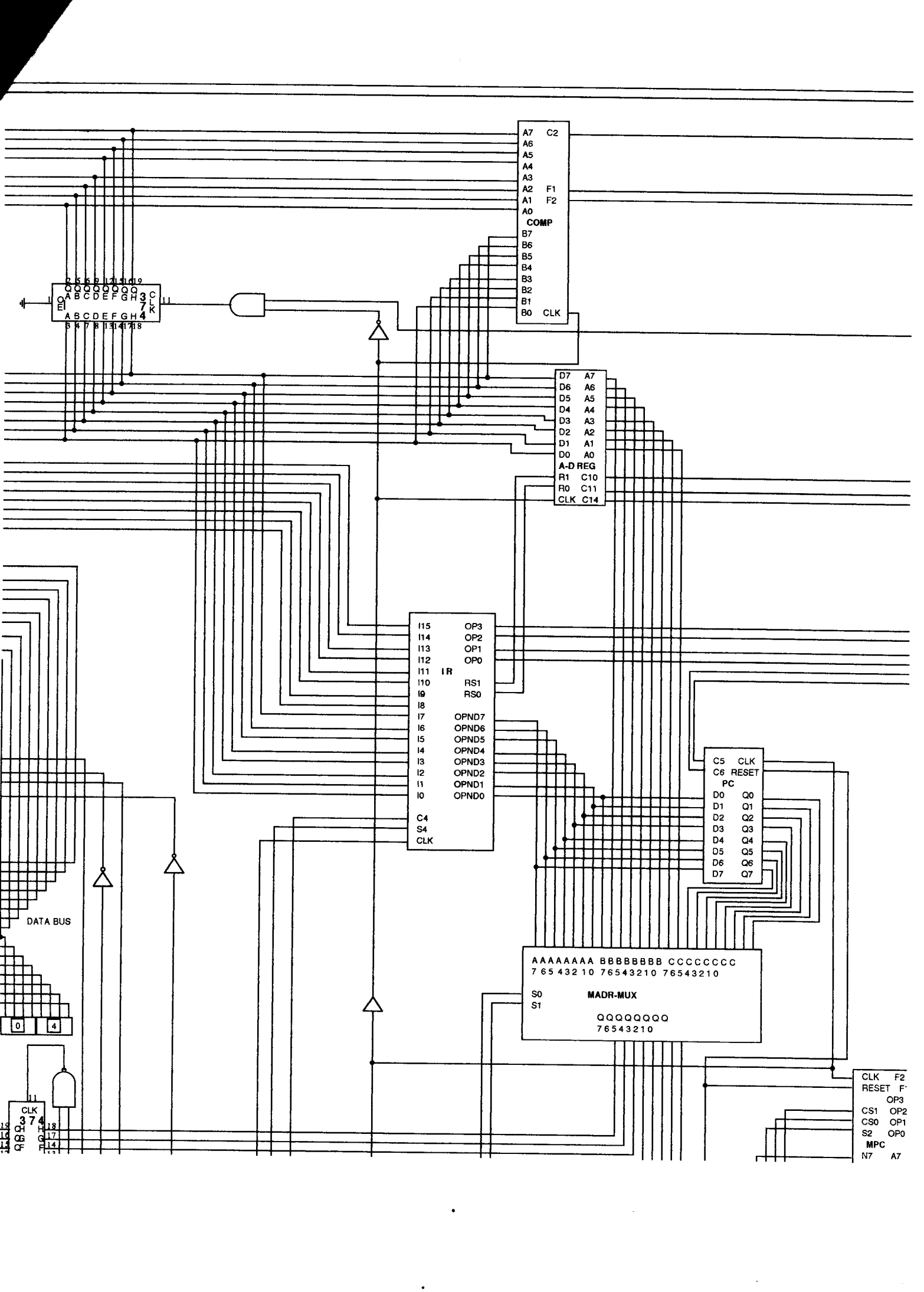
```
00    1804        MOVEI     C, 4              ; C = 9 (sort_pass_ctr)
            loop2:
01    1C40        MOVEI     D, 40H            ; D = beginning addr of data
            loop1:
02    2300        LOAD      A, [D]            ; A = *D
03    4C00        INC       D                 ; D++
04    2700        LOAD      B, [D]                  ; B = *D
05    5C00        DEC       D                 ; D--
06    6100        COMPR     A, B              ; if (A >= B) {      /* SWAP */
07    800C        JLT       skip              ;
08    4C00        INC       D                 ;       D++
09    3300        STORE     [D], A            ;       *D = A
0A    5C00        DEC       D                 ;       D--
0B    3700        STORE     [D], B                  ;       *D = B
            skip:                             ; }                  /* END */
0C    4C00        INC       D                 ; D++
0D    1044        MOVEI     A, 44H            ; A = ending addr of data
0E    6C00        COMPR     D, A              ; if (D < ending address)
0F    8002        JLT       loop1             ;       goto loop1
10    5800        DEC       C                 ; sort_pass_ctr--
11    1000        MOVEI     A, 0              ; A = 0
12    6800        COMPR     C, A              ; if (sort_pass_ctr > 0)
13    7001        JGT       loop2             ;       goto loop2

            ;     *** display results ***
14    1C40        MOVEI     D, 40H            ; D = beginning addr of data
15    1845        MOVEI     C, 45H            ; C = ending addr of data + 1
            print_loop:
16    2300        LOAD      A, [D]            ; A = *D
17    1480        MOVEI     B, 80H
18    3100        STORE     [B], A            ; putch(A);
19    4C00        INC       D                 ; D++
1A    6E00        COMPR     D, C              ; if (D <= ending data addr)
1B    8016        JLT       print_loop        ;       goto print_loop
1C    9000        HALT
```

This program sorts five values in memory locations 40 -> 44H. Due to the simulation speed on the MacIntosh, this program takes about 20 minutes to execute in *Logicworks*. Except for excessive execution time, the CPU and program are not limited to only five values.

CPU Project
Kim Berry and Miwa Nakamura

This CPU will sort
numbers and display on OUT register




A7 C2
A6
A5
A4
A3
A2 F1
A1 F2
A0
COMP
B7
B6
B5
B4
B3
B2
B1
B0 CLK
D7 A7
D6 A6
D5 A5
D4 A4
D3 A3
D2 A2
D1 A1
D0 A0
A-D REG
R1 C10
R0 C11
CLK C14
I15 OP3
I14 OP2
I13 OP1
I12 OP0
I11 IR
I10 RS1
I9 RS0
I8
I7 OPND7
I6 OPND6
I5 OPND5
I4 OPND4
I3 OPND3
I2 OPND2
I1 OPND1
I0 OPND0
C4
S4
CLK
C5 CLK
C6 RESET
PC
D0 Q0
D1 Q1
D2 Q2
D3 Q3
D4 Q4
D5 Q5
D6 Q6
D7 Q7
DATA BUS
AAAAAAAA BBBBBBBB CCCCCCCC
7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0
S0
S1
MADR-MUX
QQQQQQQQ
7 6 5 4 3 2 1 0
CLK F2
RESET F
OP3
CS1 OP2
CS0 OP1
S2 OP0
MPC
N7 A7