Operating System and Virtual Memory

Computer Science 240

Laboratory 12

Operating System

Can execute an operating system loop and service 3 processes. Each process is allowed to execute for 16 instructions before it is moved to the tail of the queue.

Operating system always uses address 0 – 63, PID 1 uses 64 – 127, PID 2 uses 128 – 191, and PID 3 uses 191 – 255. The appropriate section of memory is accessed by using the PID as the two most significant bits of the address throughout the datapath.

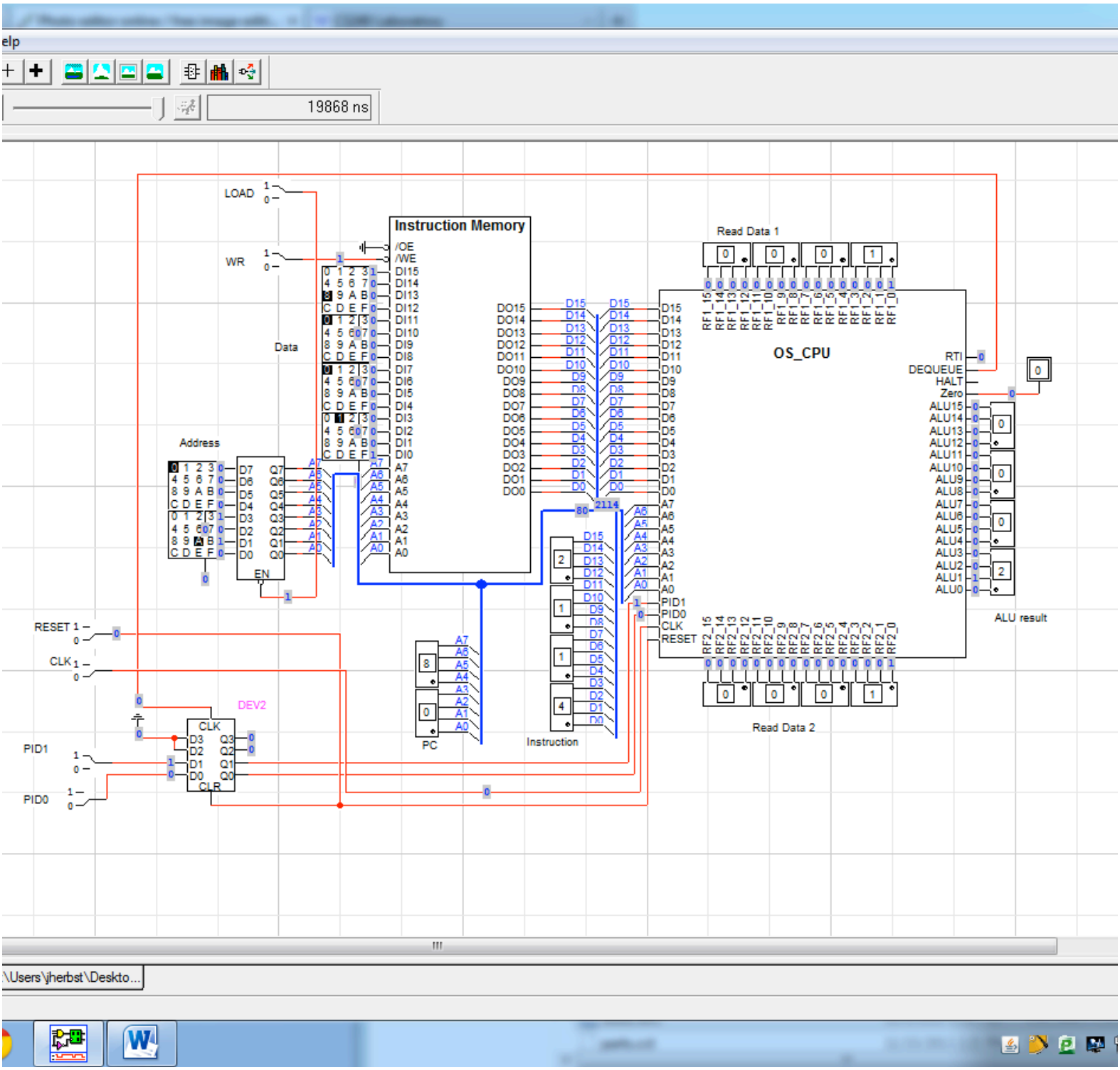
PC and the register file have been modified so that the state is saved in duplicate registers. The appropriate devices are selected by the PID.

New instructions:

HALT is actually a user instruction, but is an important addition to the basic instruction set, because it allows a graceful exit from a program.

DEQUEUE accepts a new PID from the process queue into the processor and stores the PID at address 0 in data memory.

RTI transfers control of the processor (PC, registers, and memory) to the process, through use of the PID.


elp
19868 ns
LOAD
WR
Instruction Memory
/OE
/WE
DI15
DI14
DI13
DI12
DI11
DI10
DI9
DI8
DI7
DI6
DI5
DI4
DI3
DI2
DI1
DI0
Data
Address
D7 Q7
D6 Q6
D5 Q5
D4 Q4
D3 Q3
D2 Q2
D1 Q1
D0 Q0
EN
A7
A6
A5
A4
A3
A2
A1
A0
DO15
DO14
DO13
DO12
DO11
DO10
DO9
DO8
DO7
DO6
DO5
DO4
DO3
DO2
DO1
DO0
2114
Read Data 1
OS_CPU
RTI
DEQUEUE
HALT
Zero
ALU15
ALU14
ALU13
ALU12
ALU11
ALU10
ALU9
ALU8
ALU7
ALU6
ALU5
ALU4
ALU3
ALU2
ALU1
ALU0
ALU result
PID1
PID0
CLK
RESET
Read Data 2
RESET
CLK
DEV2
CLK
D3 Q3
D2 Q2
D1 Q1
D0 Q0
CLR
PID1
PID0
PC
Instruction
\Users\jherbst\Deskto...

Mini-MIPS Instructions

Opcode	Instruction	Operands
0000	LW	Rs Rt offset
0001	SW	Rs Rt offset
0010	ADD	Rs Rt Rd
0011	SUB	Rs Rt Rd
0100	AND	Rs Rt Rd
0101	OR	Rs Rt Rd
0110	SLT	Rs Rt Rd
0111	BEQ	Rs Rt offset
1000	JMP	12-bit offset

JMP has extra bits which can be used to encode additional instructions:

1000	0000	JMP
1000	1000	DEQUEUE
1000	1100	RTI
1000	1111	HALT

JMP doesn't use the regular datapath, so addition of these instructions doesn't require too many modifications.

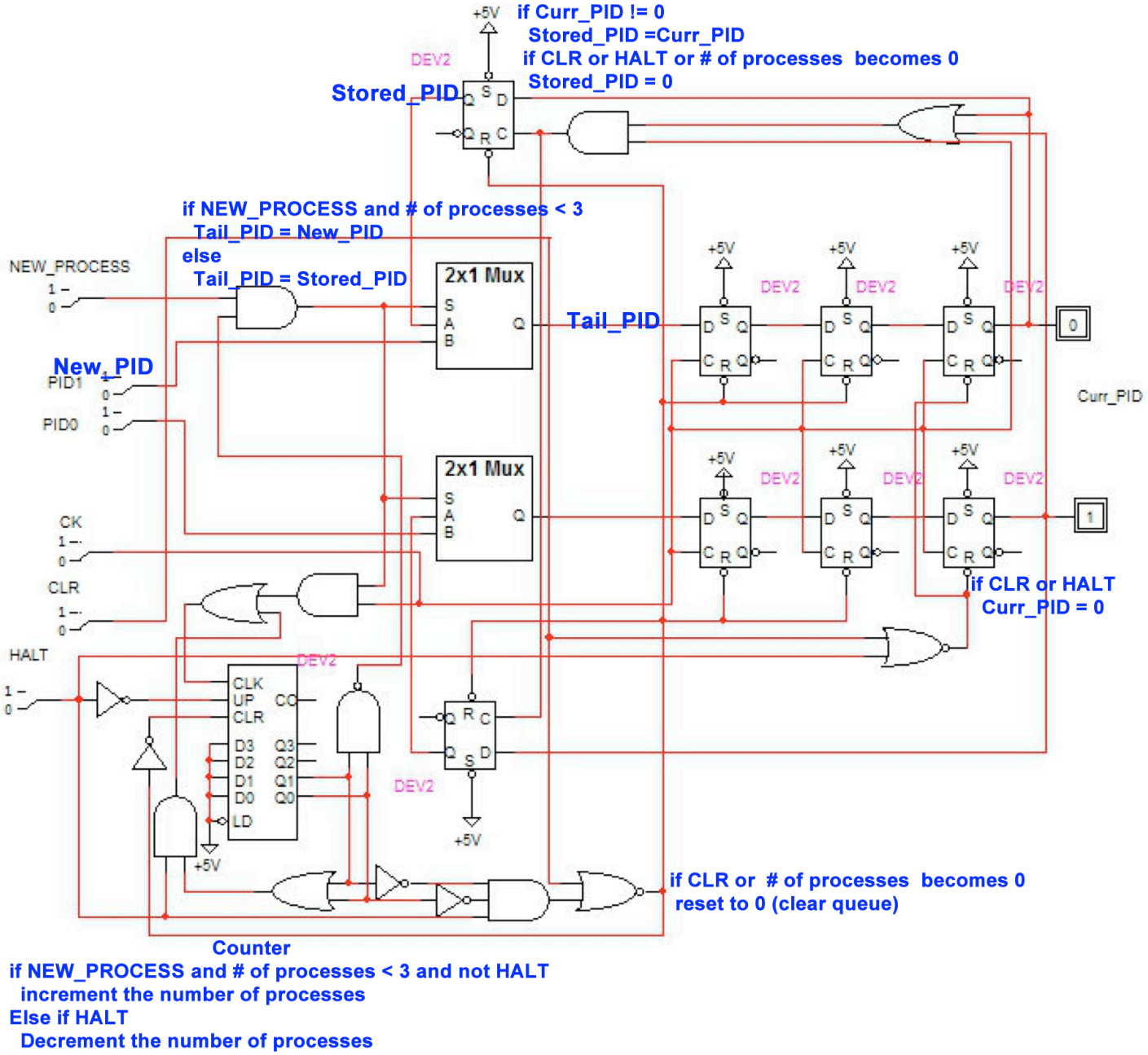
Operating System Code

Loop which is always executing:

Addr	Instruction	
00	JMP 0 0 1	# loop actually starts at addr 2,
02	DEQUEUE	# get current PID and stores in address 0 of data memory
04	LW R0 R2 0	# test PID to see if it is non-zero
06	BEQ R0 R2 0	# if 0, no processes in queue, keep looping until one shows up
08	RTI	# transfer control of the processor to the process, using the PID
0A	JMP 0 0 1	# repeat the loop when the process has executed for 16 clock cycles

Process ID Queue

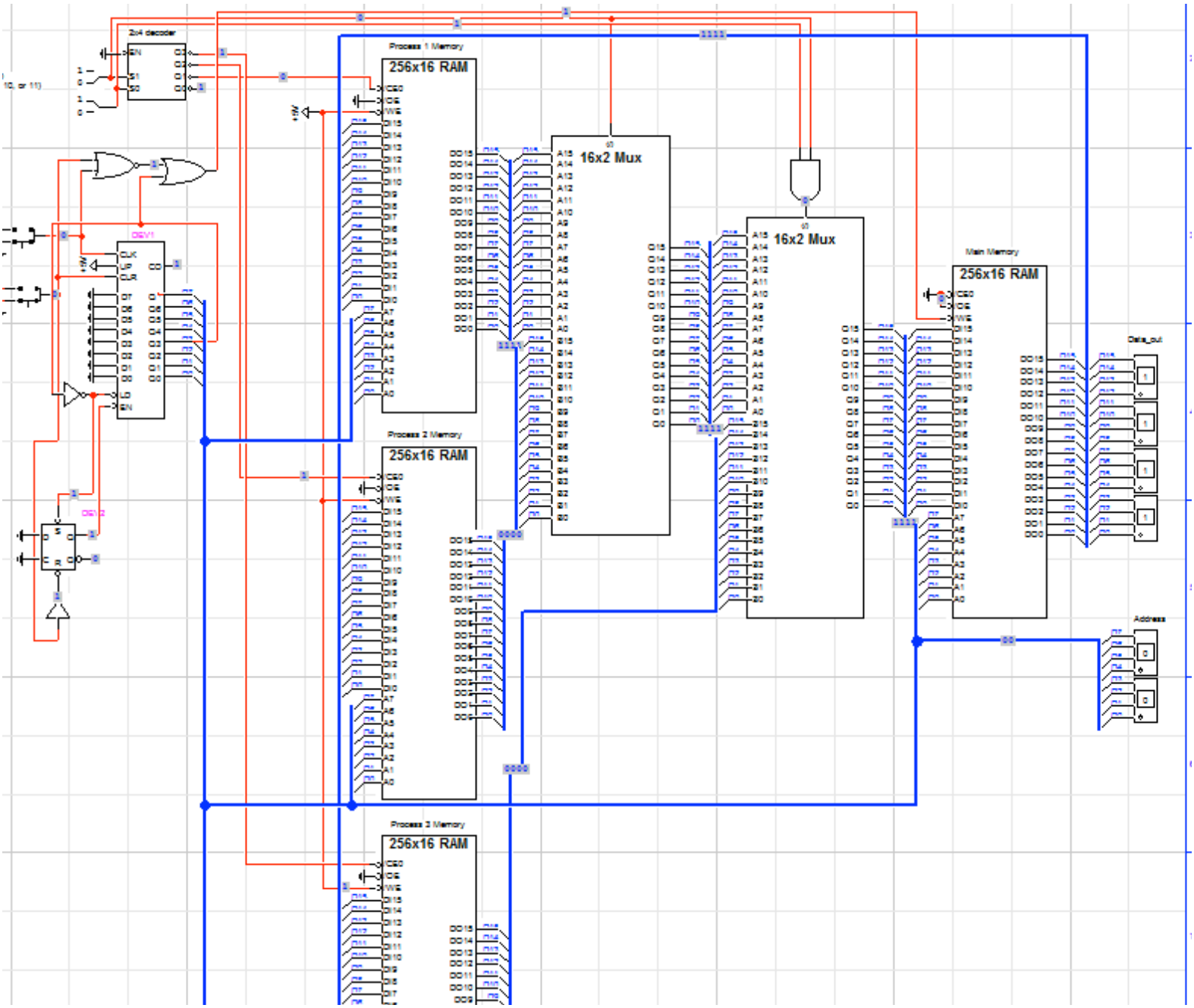
Allows up to three processes to be scheduled



Virtual Memory

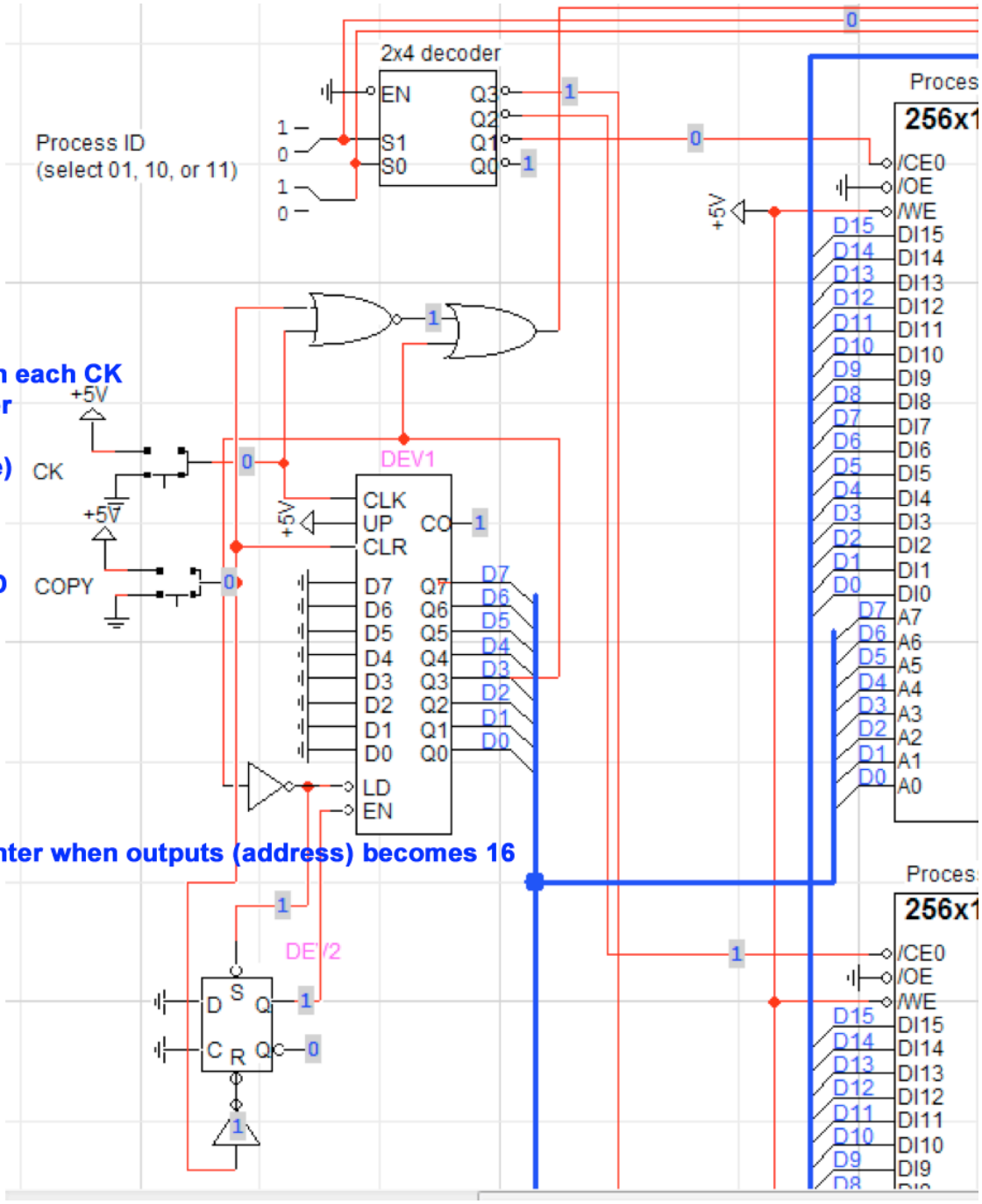
The circuit will select a memory device based on the *process id*, or PID (which can be 1, 2, or 3), and will copy the contents of the device to the main memory, so that the process can be executed.

For simplicity, only 8 locations are copied from each device (although this could easily be extended to include the entire address space).



Write (copy) to memory on each CK and increment the counter to the next addresss unless address = 16 (done)

Start a new COPY to memory of a selected PID



Disable counter when outputs (address) becomes 16