Let us now look at some of the full microinstruction sequences for these instructions:

Control store address	Next Address	JAM	ALU	C bus	Memory	B bus	hex
Main Loop (fetch)							
100	000000000	100	00110101	000000100	001	0001	004350211
NOP							
000	100000000	000	00000000	000000000	000	0000	800000000
ILOAD							
015	000010110	000					
016	000010111	000					
017	000011000	000					
018	000011001	000					
019	100000000	000	00010100	001000000	000	0000	800142000
IAND							
07E	001111111	000					
07F	011100000	000					
0E0	100000000	000					
GOTO							
0A7	010101000	000					
0A8	010101001	000					
0A9	010101010	000					
0AA	010101011	000					
0AB	011010000	000					
0D0	100000000	000	00000000	000000000	000	0000	800000000
IFLT							
09B	010011100	000					
09C	010011101	000					
09D	010011110	000					
09E	011110000	010	00010100	000000000	000	1000	782140008
1F0	010101000	000					
F0	011110001	000					
F1	011110010	000					
F2	100000000	000	00000000	000000000	000	0000	800000000
WIDE							
C4	100000000	100	00110101	000000100	001	0001	804350211
WIDE_ILOAD							
115	100010110	000					
116	100010111	000					
117	100011000	000					
118	000010111	000					

The final schematic for our cpu is given on page 254 of your textbook. There are two features of this code we need to consider:

- The box labeled "high bit" contains combinational circuitry that takes as input the values of the JAMN and JAMZ bits of the microinstruction, the value of the μ_8 bit of the microinstruction (bit 35), and the values of the N and Z flip-flops. It produces the following output value F, which will become the value of the leftmost bit of the MPC register:

$$F = (JAMZ \bullet Z) + (JAMN \bullet N) + \mu_8$$

- The box labeled OR contains combinational circuitry that takes as input the 8 bits of the MBR, the rightmost 8 bits of the next address field (that is bits $\mu_7\mu_6\mu_5\mu_4\mu_3\mu_2\mu_1\mu_0$), and the value of the JMPC bit. If JMPC = 1 then the box produces as output the (8-bit) bitwise-or of the MBR bits with$_7\mu_6\mu_5\mu_4\mu_3\mu_2\mu_1\mu_0$; if JMPC = 0 then the box merely passes the value of $\mu_7\mu_6\mu_5\mu_4\mu_3\mu_2\mu_1\mu_0$ through to the rightmost 8 bits of MPC.

Section 4. A Hardwired Control Unit

The cpu that we designed in the previous section is an example of what is known as a ***microprogrammed control unit*** because of the use of microinstructions and the microprogram storage unit. There is, however and alternative approach to implementing the control unit known as a ***hardwired control unit***. We indicate the overall approach for such a control unit here. This approach is especially useful for machine architectures that do not have a large number of machine instructions, or instructions as complex as INVOKEVIRTUAL.

With a hardwired control unit the correct control signals for each register transfer step of each phase of the instruction cycle are generated at the proper time by means of a clock-driven counter interfaced with a decoder as shown below.

Each output line of this sequence corresponds to a step number in a register-transfer sequence, where 2^n is equal to, or greater than, the maximum number of steps that will ever be required..

The required control signals to be activated for each register transfer will then be generated by additional ***combinational*** circuitry that uses the following input signals:

a. The output of the step-sequences, so we know what step we are on.
b. The content of the MBR at the end of fetch, so we know which instruction we are executing
c. The values of any condition codes (such as the N and Z signals) that tell us whether the result of an alu operation was negative, or zero, for example.

In our examples at the end of Section 2. of this chapter we derived some of the sequences of control signals needed to implement the fetch phase of the instruction cycle as well as each machine language instruction. We note that at most 23 steps are needed to carry out any instruction (1 for fetching it, and at most 22 to implement it). If we let T0,...,T22 denote the first 23 output lines of the counter-decoder circuit given above, representing lines that are active (i.e. = 1) on the successive clock ticks, then we can specify the values that the control signals must have to correctly implement the instruction cycle for our machine language instruction set by control equations such as the following

```
fetch =


 mbru =
```