1. As described in Section 3.7.2, the IA32 instruction **'leave'** can be used to prepare the stack for returning. It is equivalent to the following Y86 code sequence:

```
1 rrmovl %ebp, %esp      Set stack pointer to beginning of frame
2 popl %ebp              Restore saved %ebp and set stack ptr to end of caller's frame
```

Suppose we add this instruction to the Y86 instruction set, using the following encoding:

Byte	0	1	2	3	4	5
Leave	D	0				

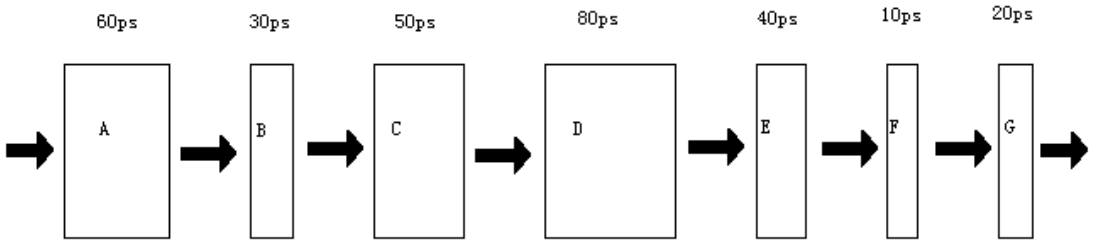
Describe the computations performed to implement this instruction. Use the computations for popl (Figure 4.20) as a guide.

2. Consider executing the following code on PIPE (solving data hazards with forwarding):

```
addl      %ebx, %eax
mrmovl    10(%eax), %ecx
mrmovl    20(%ecx), %ecx
addl      %ecx, %ebx
```

Identify all of the data hazards in the above code.

3. Suppose we have a sequence of combinational logic of Figure and called A to G. The delays are marked upper them respectively.



Create a pipelined version of this design by inserting 4 registers between pairs of these blocks to get the maximum throughput arise. Draw the design graph and calculate the throughput and latency. Delay of each register is 10ps.

4. In SEQ fetch stage, the signal instr_valid indicates whether an instruction is legal or not. Write HCL code for instr_valid.

5. The register signal srcB indicates which register should be read to generate the signal valB. The desired value is shown as the second step in the decode stage in Figures 4.18 to 4.21. Write HCL code for srcB.