IT5002 Computer Systems and Applications
Tutorial 4

1. Suppose the five stages in some 5-stage pipeline (fetch-decode-ALU-memory-writeback) take the following timing: 2ns, 3ns, 4ns, 8ns and 2ns. Given 1000 instructions, what is the speedup of running on a pipelined processor versus:

   i. A single-cycle non-pipelined CPU.

   ii. A multi-cycle non-pipelined CPU, given that 70% of the instructions are arithmetic, 25% are branch instructions, and 2% are loads and 3% are stores.

2. Let's try to understand pipeline processor by doing a detailed trace. Suppose the pipeline registers (also known as pipeline latches) store the following information:

IF / ID	
No Control Signal	------
PC+4	
OpCode	
Rs	
Rt	
Rd	
Funct	
Imm(16)	

ID / EX	
MToR	
RegWr	
MemRd	
MemWr	
Branch	
RegDst	
ALUsrc	
ALUop	
PC+4	
ALUOpr1	
ALUOpr2	
Rt	
Rd	
Imm(32)	

EX / MEM	
MToR	
RegWr	
MemRd	
MemWr	
Branch	
BrcTgt	
isZero?	
ALURes	
ALUOpr2	
DstRNum	

MEM / WB	
MToR	
RegWr	
MemRes	
ALURes	
DstRNum	

Show the progress of the following instructions through the pipeline stages by filling in the content of pipeline registers.

```
i.   0x8df80000  # lw $24, 0($15)     #Inst.Addr = 0x100
ii.  0x1023000C  # beq $1, $3, 12     #Inst.Addr = 0x100
iii. 0x0285c822  # sub $25, $20, $5   #Inst.Addr = 0x100
```

Assume that registers 1 to 31 have been initialized to a value that is equal to 101 + its register number. i.e. [$1] = 102, [$31] = 132 etc. You can put "X" in fields that are irrelevant for that instruction. Do note that in reality, these fields are actually generated but not utilized.

Part (i) has been worked out for you.

i. `0x8df80000 # lw $24, 0($15) #Inst.Addr = 0x100`

IF / ID	
No Control Signal	------
PC+4	0x104
OpCode	0x23
Rs	$15
Rt	$24
Rd	X
Funct	X
Imm(16)	0

ID / EX	
MToR	1
RegWr	1
MemRd	1
MemWr	0
Branch	0
RegDst	0
ALUsrc	1
ALUop	00
PC+4	0x104
ALUOpr1	116
ALUOpr2	X
Rt	$24
Rd	X
Imm(32)	0

EX / MEM	
MToR	1
RegWr	1
MemRd	1
MemWr	0
Branch	0
BrcTgt	X
isZero?	X
ALURes	116
ALUOpr2	X
DstRNum	$24

MEM / WB	
MToR	1
RegWr	1
MemRes	Mem(116)
ALURes	X
DstRNum	$24

ii. `0x1023000C # beq $1, $3, 12 #Inst.Addr = 0x100`

iii. `0x0285c822 # sub $25, $20, $5 #Inst.Addr = 0x100`

3. Given the following three formulas

$$CT_{seq} = \sum_{k=1}^{N} T_k$$

$$CT_{pipeline} = \max(T_k) + T_d$$

$$Speedup_{pipeline} = \frac{CT_{seq} \times InstNum}{CT_{pipeline} \times (N + InstNum - 1)}$$

For each of the following processor parameters, calculate CT_{seq}, $CT_{pipeline}$ and $Speedup_{pipeline}$ (to two decimal places) for 10 instructions and for 10 million instructions.

	Stages Timing (for 5 stages, in ps)	Latency of pipeline register (in ps)
a.	300, 100, 200, 300, 100 (slow memory)	0
b.	200, 200, 200, 200, 200	40
c.	200, 200, 200, 200, 200 (ideal)	0

4. Here is a series of address references in decimal: 4, 16, 32, 20, 80, 68, 76, 224, 36, 44, 16, 172, 20, 24, 36, and 68 in a MIPS machine. Assuming a **direct-mapped cache** with 16 one-word blocks that is initially empty, label each address reference as a hit or miss and show the content of the cache.

   You may write the data word starting at memory address X as M[X]. (For example, data word starting at memory address 12 is written as M[12]. This implies that the word includes the 4 bytes of data at addresses 12, 13, 14 and 15.) You may write the tag values as decimal numbers. If a block is replaced in the cache, cross out the corresponding content in the cache, and write the new content over it.