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:

   In a pipelined CPU, the number of cycles taken can be calculated from:

   1. It takes 4 cycles for the first instruction to complete, since it is a 4-stage pipeline and each stage takes one cycle.

   2. It takes 1 cycle each for the remaining N-1 instructions.

   3. Total cycles = 4 + N – 1 = 4 + 999 = 1003 cycles.

   4. The cycle time must be the slowest of all the stages (otherwise the slowest stage will not have time to complete) = 8ns

   5. Total time = 8024 ns

      i. A single-cycle non-pipelined CPU.

   Cycle time must be equal to the sum of the time of all the stages = 2 + 3 + 4 + 8 + 2 = 19ns.
   Total time = 1000 x 19 = 19000 ns.
   Speedup = 19000 / 8024 = 2.37 times.

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.

We need to first work out the instruction times for each type of instruction based on the stages they use. Each stage must take 8ns, the timing of the slowest stage.

Instruction	IF 2ns	ID 3ns	ALU 4ns	MEM 8ns	WB 2ns	Time ns
Arithmetic	X	X	X		X	$4 \times 8 = 32$
Branch	X	X	X			$3 \times 8 = 24$
Load	X	X	X	X	X	$5 \times 8 = 40$
Store	X	X	X	X		$4 \times 8 = 32$

Total cycle time = $700 \times 32 + 250 \times 24 + 20 \times 40 + 30 \times 32$
= 22,400 + 6000 + 800 + 960
= 30,160 ns

Speedup = 30,160 / 8024 = 3.76 times.

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

Answers:

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

IF / ID	
No Control Signal	------
PC+4	0x104
OpCode	4
Rs	$1
Rt	$3
Rd	X
Funct	X
Imm(16)	12

ID / EX	
MToR	X
RegWr	0
MemRd	0
MemWr	0
Branch	1
RegDst	X
ALUsrc	0
ALUop	01
PC+4	0x104
ALUOpr1	102
ALUOpr2	104
Rt	X
Rd	X
Imm(32)	12

EX / MEM	
MToR	X
RegWr	0
MemRd	0
MemWr	0
Branch	1
BrcTgt	0x134
isZero?	0
ALURes	-2
ALUOpr2	X
DstRNum	X

MEM / WB	
MToR	X
RegWr	0
MemRes	X
ALURes	X
DstRNum	X

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

IF / ID	
No Control Signal	------
PC+4	0x104
OpCode	0
Rs	$20
Rt	$5
Rd	$25
Funct	22
Imm(16)	X

ID / EX	
MToR	0
RegWr	1
MemRd	0
MemWr	0
Branch	0
RegDst	1
ALUsrc	0
ALUop	10
PC+4	0x104
ALUOpr1	121
ALUOpr2	106
Rt	X
Rd	$25
Imm(32)	X

EX / MEM	
MToR	0
RegWr	1
MemRd	0
MemWr	0
Branch	0
BrcTgt	X
isZero?	X
ALURes	15
ALUOpr2	X
DstRNum	$25

MEM / WB	
MToR	0
RegWr	1
MemRes	X
ALURes	15
DstRNum	$25

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

Answers:

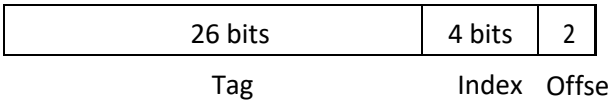
	CT_{seq}	**$CT_{pipeline}$**	**Speedup (10 inst)**	**Speedup (10m inst)**
a.	**1000ps**	**300ps**	(1000×10)/(300×14) **= 2.38**	(1000×10m)/(300 (10m+4)) **= 3.33**
b.	**1000ps**	**240ps**	(1000×10)/(240×14) **= 2.98**	(1000×10m)/(240×(10m+4)) **= 4.17**
c.	**1000ps**	**200ps**	(1000×10)/(200×14) **= 3.57**	(1000×10m)/(200×(10m+4)) **= 5.00**

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, dataword 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.

Answer:

Since this is a MIPS machine, a word consists of 4 bytes. Should first work out the tag, index, and offset fields:



4:	00…00	0001	00 ← Miss
16:	00…00	0100	00 ← Miss
32:	00…00	1000	00 ← Miss
20:	00…00	0101	00 ← Miss
80:	00…01	0100	00 ← Miss
68:	00…01	0001	00 ← Miss
76:	00…01	0011	00 ← Miss
224:	00…11	1000	00 ← Miss
36:	00…00	1001	00 ← Miss
44:	00…00	1011	00 ← Miss
16:	00…00	0100	00 ← Miss
172:	00…10	1011	00 ← Miss
20:	00…00	0101	00 ← Hit
24:	00…00	0110	00 ← Miss
36:	00…00	1001	00 ← Hit
68:	00…01	0001	00 ← Hit

Cache block	Valid bit	Tag	Word
0	0		
1	~~0~~ 1	~~0~~ 1	~~M[4]~~ M[68]
2	0		
3	~~0~~ 1	1	M[76]
4	~~0~~ 1	~~0 1~~ 0	~~M[16] M[80]~~ M[16]
5	~~0~~ 1	0	M[20]
6	~~0~~ 1	0	M[24]
7	0		
8	~~0~~ 1	~~0~~ 3	~~M[32]~~ M[224]
9	~~0~~ 1	0	M[36]
10	0		
11	~~0~~ 1	~~0~~ 2	~~M[44]~~ M[172]
12	0		
13	0		
14	0		
15	0		