CSCI 2021, Spring 2015

Homework Assignment III: Solutions

<u>Problem 1:</u>

A.

					<- buf[0] ->				<- buf[1] ->				<- %ebp ->				<- ret addr ->				
					64	72	2e	65	76	69	6c	00									

4 byte integer is: 0x006c6976

B.

Before input:

					<- buf[0] ->				<- buf[1] ->				<- %ebp ->				<- ret addr ->				
													?	?	?	?	2c	84	04	08	

After input:

					<- buf[0] ->				<- buf[1] ->				<- %ebp ->				<- ret addr ->				
					64	72	2e	65	76	69	6c	2e	6c	69	76	65	73	00	04	08	

a.

buf[0] = 0x652e7264

buf[3] = 0x08040073

b.

%ebp = 0x6576696c

Problem 2:

A.

Stack object	Address of stack object
return address	&buf[[0] + 12
old %ebp	&buf[0] + 8
buf[3]	&buf[0] + 3
buf[2]	&buf[0] + 2
buf[1]	&buf[0] + 1
buf[0]	&buf[0] + 0

B.

Answer: "`0 0 0 0 0 0 0 0 0 0 0 0 71 85 04 08`"

Problem 3:

leave:

Fetch	icode:ifun <- M1[PC]; valP <- PC + 1;
Decode	valA <- R[%ebp];
Execute	valE <- valA + 4;
Memory	valM <- M4[valA];
Writeback	R[%esp] <- valE; R[%ebp] <- valM;
PC Update	PC <- valP;

iaddl c, rb:

Fetch	icode:ifun <- M1[PC]; ra:rb <- M1[PC+1]; valC <- M4[PC+2]; valP <- PC + 6;
Decode	valB <- R[rb];
Execute	ValE <- valB + valC;
Memory	
Writeback	R[rb] <- valE;
PC Update	PC <- valP;

jmp c(ra):

Fetch	icode:ifun <- M1[PC]; ra:rb <- M1[PC+1]; valC <- M4[PC+2]; valP <- PC + 6;
Decode	valA <- R[ra];
Execute	valE <- valA + valC;
Memory	valM <- M4[valE];
Writeback	
PC Update	PC <- valM;

Problem 4:

Iteration 1																				
Instr	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Iteration 1																				
In 1	F	D	**E**	M	W															
In 2		F	D	D	D	D	**E**	M	W											
In 3			F	F	F	F	D	**E**	M	W										
In 4							F	D	D	D	D	**E**	M	W						
In 5								F	F	F	F	D	**E**	M	W					
In 6												F	D	**E**	M	W				
In 7													F	D	D	**E**	M	W		
In 8														F	F	D	**E**	M	W	
Iteration 2																				
In 1	M	W																F	D	**E**
In 2	D	D	D	**E**	M	W													F	D
In 3	F	F	F	D	**E**	M	W													F
In 4				F	D	D	D	D	**E**	M	W									
In 5					F	F	F	F	D	**E**	M	W								
In 6									F	D	**E**	M	W							
In 7										F	D	D	**E**	M	W					
In 8											F	F	D	**E**	M	W				
Iteration 3																				
In 1															F	D	**E**	M	W	
In 2	**E**	M	W													F	D	D	D	D
In 3	D	**E**	M	W													F	F	F	F
In 4	F	D	D	D	D	**E**	M	W												
In 5		F	F	F	F	D	**E**	M	W											
In 6						F	D	**E**	M	W										
In 7							F	D	D	**E**	M	W								
In 8								F	F	D	**E**	M	W							

[1] CPI = 1.0 + $\frac{Number\ of\ bubbles}{Number\ of\ instructions\ or\ Number\ of\ Executes}$ = 1.0 + $\frac{25}{24}$ $\approx$ 2.042

[2] If a clock operates at 1GHz, Then each cycle takes 10^{-9} seconds.

Now, given the CPI of 2.042, we expect an execute every 2.042 cycles or (2.042 * 10^{-9}) seconds

Therefore the instruction per second achieved = $\frac{1\ instruction}{(2.042 *\ 10^{-9})\ seconds}$ = 0.49 GIPS = 490 MIPS.

GIPS: Giga instructions per second.

MIPS: Mega instructions per second.

Problem 5:

Iteration 1																				
Instr	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Iteration 1																				
In 1	F	D	*E*	***M***	W															
In 2		F	D	***D***	E	M	W													
In 3				F	D	***E***	***M***	***W***												
In 4					F	***D***	E	M	***W***											
In 5						F	***D***	E	M	W										
In 6							F	***D***	E	M	W									
In 7								F	***D***	E	M	W								
In 8									F	D	E	M	W							
Iteration 2																				
In 1										F	D	*E*	***M***	W						
In 2											F	D	***D***	E	M	W				
In 3													F	D	***E***	***M***	***W***			
In 4														F	***D***	E	M	***W***		
In 5															F	***D***	E	M	W	
In 6																F	***D***	E	M	W
In 7	W																F	***D***	E	M
In 8	M	W																F	D	E
Iteration 3																				
In 1	*E*	***M***	W																F	D
In 2	D	***D***	E	M	W															F
In 3		F	D	***E***	***M***	***W***														
In 4			F	***D***	E	M	***W***													
In 5				F	***D***	E	M	W												
In 6					F	***D***	E	M	W											
In 7						F	***D***	E	M	W										
In 8							F	D	E	M	W									

[1] CPI = 1.0 + $\frac{Number\ of\ bubbles}{Number\ of\ instructions\ or\ Number\ of\ Executes}$ = 1.0 + $\frac{3}{24}$ ≈ 1.125

[2] If a clock operates at 1GHz, Then each cycle takes 10^{-9} seconds.

Now, given the CPI of 1.125, we expect an execute every 1.125 cycles or (1.125 * 10^{-9}) seconds

Therefore the instruction per second achieved = $\frac{1\ instruction}{(1.125 * 10^{-9})\ seconds}$ = 0.889 GIPS = 889 MIPS.

Gray shading shows forwarding

Problem 6:

ExCmp would be used to indicate the completion of an ALU computation during the execute phase. Making 'execute' a 4 or 6 cycle phase because of multiplication or division would slow down all other instructions.

Therefore by introducing ExCmp, we can check whether to move the instruction forward in the pipeline.

If **ExCmp is set**, move the instruction in execute, to memory stage, move prior instructions forward in the pipeline and fetch a new instruction.

If **ExCmp is not set**, ***stall*** current instruction in Execute phase; inject a ***bubble*** in memory phase; prior instructions (in Decode and Fetch) remain in their current phases; and no new instruction is fetched.