CSE 112:
Computer Organization
Sujay Deb

End Semester Question Paper

IIIT-Delhi, Winter 2025
Duration : 120 Minutes
Date: Apr 28, 2025
Points: 80

Guidelines

*If you found any ambiguity in any of the questions or there appears to be a lack of information, then write an **logical** assumption on the answer sheet to explain your side interpretation of the problem and solve accordingly.*

First Block Instructions	Description	Operation
addi rd,rs, imm[11:0]	Add immediate	rd = rs + sext(imm[11:0])
sub rd, rs1, rs2	Subtract registers	rd = rs1 - rs2
add rd, rs1, rs2	Add registers	rd = rs1 + rs2
xor rd, rs1, rs2	Perform XOR boolean operation	rd = rs1 xor rs2
slt rd, rs1, rs2	Set Less than	rd = 1. If signed(rs1) < signed(rs2) else rd=0
sltu rd,rs1,rs2	Set Less than unsigned	rd = 1. If unsigned(rs1) < unsigned(rs2) else rd=0
lw rd,#imm[11:0](rs1)	Load Word	rd=mem(sext(rs1+imm))
sw rd,#imm[11:0](rs1)	Store Word	mem(sext(rs1+imm))=rd
beq rs1, rs2, imm[12:1]	Branch if equal	PC = PC + sext({imm[12:1],1'b0}) if rs1 == rs2
jal rd, imm[20:1]	Jump and Link	rd=PC+4. PC=PC+sext({imm[20:1], 1'b0})
beq x0,x0,#0	Branch if equal	**HALT**

Second Block Instructions	Description	Operation
Jalr ra, rs1, imm[11:0]	Jump and Link register	ra=PC+4, PC=rs1+sext(imm[11:0])

[CO2, CO3] Problem I : Pipeline **[13 Points]**

Draw the pipeline diagram of the below mentioned assembly program executed on a five stage risc-v processor with the microarchitecture described in figure[1]. You need to stall the pipeline in case of any hazard. Assume that there is always a cache hit and the instruction cache is different from the data cache.

Assembly Program

```
            Addi x2,x0,#2
            Addi x1,x0,#1
            Addi x3,x0,#256
            Lw x3,8(x3)
            Add x4,x3,x1
            Bne x2,x1,NEXT_1
            Sub x3,x2,x1
            Sub x3,x3,x0
NEXT_1:     Addi x1,x0,#2
            Beq x1,x2,NEXT_2
            Sub x4,x5,x6
            addi x6,x0,#6
NEXT_2:     Beq x0,x0,#0
```

[CO3] Problem II : Microprocessor Microarchitecture **[11+5 Points]**

You need to draw the pipeline diagram and stall the pipeline in case of unresolvable hazards, when the below-mentioned assembly code is executed on the microprocessor having microarchitecture as shown in fig[1]. And also, at each positive edge(posedge) of the clock, you need to write the status(in binary) of mentioned control signals of the microprocessor while executing the program.

A Hypothetical Example of Status Signals (for layout reference)

Posedge	StallF	StallD	FlushD	FlushE	ForwardAE[1:0]	ALUSrcE
1st	1	0	0	0	01	0

Assembly Program

```
        Addi x2,x0,#2
        Addi x1,x0,#2
        Addi x8,x0,#256
        Lw x4,8(x8)
        Add x5,x4,x1
        Add x6,x5,x1
        Beq x1,x2,LABEL
        Add x7,x1,x2
        Add x7,x1,x2
LABEL:  Sub x7, x2,x1
        Beq x0,x0,#0
```

[CO1] Problem III : ISA Format **[01+01+01+02 Points]**

The format of an ISA is described below.. The ISA has 32 registers and each instruction size is 32-bit. A 32-bit processor follows this ISA and supports byte addressable memory access.

Opcode	rd	rs1	rs2

You need to answer the below-mentioned questions

(a) How many bits are required to represent each register.
(b) How many number of bits are required to represent each opcode
(c) How many uniques instructions can the ISA support
(d) What decimal number should be added to the Program Counter to fetch the next instruction.

[CO2] Problem IV : Assembly Encoding **[10 Points]**

Write an assembly program to print the below-mentioned pattern.

Out rs1	Print the content of register rs1.
Out “\n”	Go to next line.

Pattern

1 2 3 4 5
1 2 3 4
1 2 3
1 2
1

[CO4] Problem V : Cache Indexing **[03*3+01 Points]**

A processor having byte addressable range of 64GByte is connected to a 12-way set associative cache. The size of the block in cache is 64 Bytes and the cache size is 48KByte.

(a) Find the number of bits in tag, set index, and block offset
(b) For this cache system, if you were to place block number 64 (assume block numbers start from 0) from main memory to the cache. Which set number (assuming set number counting starts from 0) will you put it in?

[CO4] Problem VI : Cache Miss Rate **[05 Points]**

Suppose you have a cache with a hit latency of 1 cycle. When it misses, it goes to main memory, which has a latency of 105 cycles (this includes the latency required to know that the access was a miss and the latency to bring the data from main memory to cpu). If the average memory access time (in cycles) is 6.2 cycles. What is the miss rate of the cache?

[CO1, CO3] Problem VII : Processor Speedup **[05 Points]**

You have built a non-pipelined processor with a frequency of 2.5 gigahertz, and the average cycles per instruction is 4. The same processor is upgraded to a pipelined processor with five stages, but due to the internal pipeline delay, the clock speed is reduced to 2 gigahertz. Assume there are no stalls in the pipeline. What is the ratio of speedup of the non-pipelined processor to the speedup of the pipeline processor for the execution of 100 instructions?

[CO2] Problem VIII : Caller Callee **[10 Points]**

You need to convert the below-mentioned pseudo program into an assembly language program. The isa supports **First Block and Second Block instructions** as specified in the beginning of the paper.

(a) The ISA has 32 general purpose registers x0,x1,....,x31
(b) x1 is the return address register
(c) x10, x11 stores the return value from the function.
(d) The arguments to the callee are stored in registers x12, x13, …, x17.
(e) The registers from x18, x19, …, x27 are caller saved.
(f) The registers x8, x9 are callee saved.
(g) Whenever the branch and link instruction is used. The return address is stored in x1, and the program counter jumps to the specified label.

Pseudo Program

```
Void func1()    {
        a=5
        b=9
        e=11
        d=func2(a,b,c)
        f=d+a   }
Int func2(a,b,c) {
        return a+b-c    }
```

[CO3] Problem IX: Branch Penalty **[06 Points]**

Consider two processors C, D following RISC ISA and having stage description as follows..

(a) Processor C has 36 stages as F1, F2, . . F10, D1, D2, ... D9, E1, E2, . . . E9, M1, M2, . . M4, W1, W2, … W4. The branch operations are determined in the stage E6. The processor operates at 1GHz.
(b) Processor D has 92 stages as F1, F2, … F30, D1, D2, … D10, E1, E2, … E20, M1, M2, … M25, W1, W2, … W7. And, the branch operations are determined in the stage E6. The processor operates at 5GHz.

You need to calculate the Branch penalty of both the processors

(a) In terms of number of cycles.
(b) In terms of time lost in the penalty in ns.

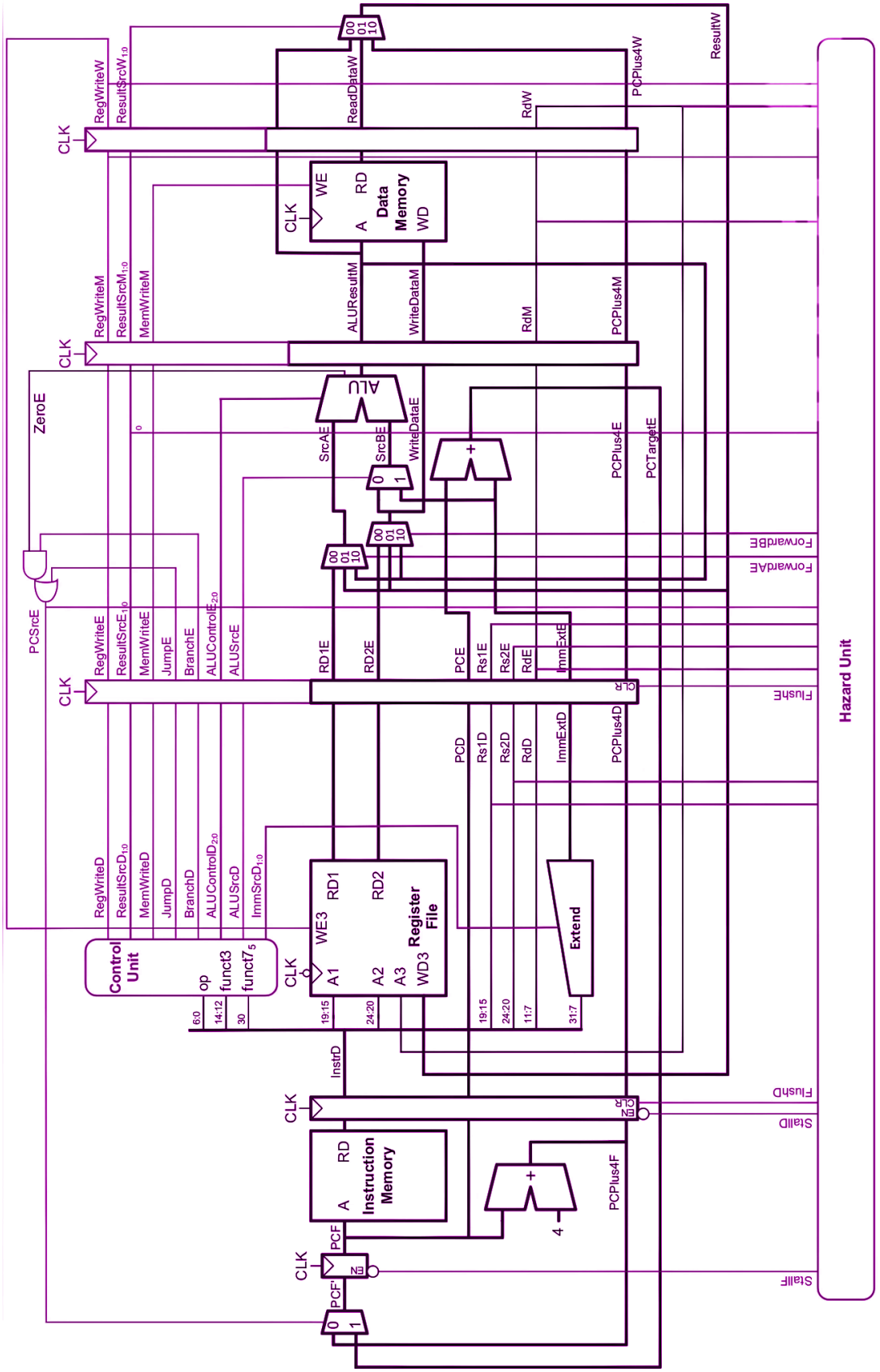

Figure 1: Microarchitecture of Five Stage RISC-V Processor following FIRST BLOCK INSTRUCTIONS. Image Source: S. L. Harris and D. Harris, "Digital Design and RISC-V Computer Architecture Textbook," *2021 ACM/IEEE Workshop on Computer Architecture Education (WCAE)*, Raleigh, NC, USA, 2021

[CO2, CO3] Problem I : Pipeline **[13 Points]**

Draw the pipeline diagram of the below mentioned assembly program executed on a five stage risc-v processor with the microarchitecture described in figure[]. You need to stall the pipeline in case of any hazard. Assume that there is always a cache hit and we have two different instructions and data cache.

Assembly Program

```
            Addi x2,x0,#2
            Addi x1,x0,#1
            Addi x3,x0,#256
            Lw x3,8(x3)
            Add x4,x3,x1
            Bne x2,x1,NEXT_1
            Sub x3,x2,x1
            Sub x3,x3,x0
NEXT_1:     Addi x1,x0,#2
            Beq x1,x2,NEXT_2
            Sub x4,x5,x6
            addi x6,x0,#6
NEXT_2:     Beq x0,x0,#0
```

Solution

	Instruction	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17		Pt
	Addi x2,x0,#2	F	D	E	M	W														1
	Addi x1,x0,#1		F	D	E	M	W													1
	Addi x3,x0,#256			F	D	E	M	W												1
	Lw x3,8(x3)				F	D	E	M	W											1
	Add x4,x3,x1					F	D	D	E	M	W									1
	Bne x2,x1,**NEXT_1**						F	F	D	E	M	W								1
	Sub x3,x2,x1								F	D	-	-								1
	Sub x3,x3,x0									F	-									1
NEXT_1:	Addi x1,x0,#2										F	D	E	M	W					1
	Beq x1,x2,**NEXT_2**											F	D	E	M	W				1
	Sub x4,x5,x6												F	D	-	-				1
	addi x6,x0,#6													F	-					1
NEXT_2:	Beq x0,x0,#0														F	D	E	M	W	1

One more possible solution for question 1.

	Instruction	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
	Addi x2,x0,#2	F	D	E	M	W																			
	Addi x1,x0,#1		F	D	E	M	W																		
	Addi x3,x0,#256			F	D	E	M	W																	
	Lw x3,8(x3)				F	D	D	D	D	E	M	W													
	Add x4,x3,x1					F	F	F	F	D	D	D	D	E	M	W									
	Bne x2,x1,**NEXT_1**									F	F	F	F	D	E	M	W								
	Sub x3,x2,x1													F	D	-	-								
	Sub x3,x3,x0														F	-	-								
NEXT_1:	Addi x1,x0,#2															F	D	E	M	W					
	Beq x1,x2,**NEXT_2**																F	D	D	D	D	E	M	W	
	Sub x4,x5,x6																	F	F	F	F	D	-	-	
	addi x6,x0,#6																					F	-	-	
NEXT_2:	Beq x0,x0,#0																						F	D	E

[CO3] Problem II : Microprocessor Microarchitecture **[11+5 Points]**

At each posedge of clock, you need to write the status(in binary) of mentioned control signals of the microprocessor fig[] while executing the below-mentioned assembly program. You also need to draw the pipeline diagram and stall in case of unresolvable hazards.

A Hypothetical Example Status

Posedge	StallF	StallD	FlushD	FlushE	ForwardAE[1:0]	ALUSrcE
1st	1	0	0	0	01	0

Assembly Program

```
        Addi x2,x0,#2
        Add x1,x0,#2
        Addi x8,x0,#256
        Lw x4,8(x8)
        Add x5,x4,x1
        Add x6,x5,x1
        Beq x1,x2,LABEL
        Add x7,x1,x2
        Add x7,x1,x2
LABEL:  Sub x7, x2,x1
        Beq x0,x0,#0
```

Solution

Pipeline Diagram

	Instruction	1	2	3	4	5	6	7	8	9	10	11	12	Pt
	Addi x2,x0,#2	F	D	E	M	W								1
	Add x1,x0,#2		F	D	E	M	W							1
	Addi x8,x0,#256			F	D	E	M	W						1
	Lw x4,8(x8)				F	D	E	M	W					1
	Add x5,x4,x4					F	D	D	E	M	W			1
	Add x6,x5,x1						F	F	D	E	M	W		1
	Beq x1,x2,**LABEL**								F	D	E	M	W	1
	Add x7,x1,x2									F	D	-	-	1

	Add x7,x1,x2										F	-	-	1
LABEL	Sub x7, x2,x1											F	D	
	Beq x0,x0,#0												F	1

Status Table

Posedge	**StallF**	**StallD**	**FlushD**	**FlushE**	**ForwardAE[1:0]**	**ALUSrcE**	**Points**
1	0	0(x)	0(x)	0(x)	00(xx no affect till)	x	0
2	0	0	0	0(x)	00(xx no affect till)	x	0.5
3	0	0	0	0	00	1	0.5
4	0	0	0	0	00	1	0.5
5	0	0	0	0	00	1	0.5
6	0	0	0	0	10	1	0.5
7	1	1	0	0	10(xx no affect)	x(no affect)	0.5
8	0	0	0	0	01	0	0.5
9	0	0	0	0	10	0	0.5
10	0	0	0	0	00	0	0.5
11	0	0	1	1	00(xx)	x(no affect)	0.5
12	0	0	0	0	00(xx)	x(no affect)	0
13	0	0	0	0	00	0	0
14	0	0	0	0	00	1	0
15	0	0	0	0	00	1	0

[CO1] Problem III : ISA Format **[01+01+01+02 Points]**

The format of an ISA is described below.. The ISA has 32 registers and each instruction size is 32-bit. A 32-bit processor follows this ISA and supports byte addressable memory

access.

Opcode	rd	rs1	rs2

You need to answer the below-mentioned questions

(e) How many bits are required to represent each register.
(f) How many number of bits are required to represent each opcode
(g) How many uniques instructions can the ISA support
(h) What number should be added to the Program Counter to fetch the next instruction.

Solution

(a) Total registers = 32
Number of bits required to represent each register = $\log_2(32) = 5$ [01 Point]
(b) Instruction size = 32
Number of bits required to represent the opcode = 32 -3*5 = 17 [01 Point]
(c) Number of unique instructions = 2^17 = 131072 [01 Point]
(d) Our instruction size is 32-bit or 4-byte. As, the processor is byte addressable. We need to jump by 4. Or the Program counter will be incremented by 4 every time. [02 Point]

[CO] Problem IV : Assembly Encoding [10 Points]

Write an assembly program to print the below-mentioned pattern.

Pattern

1 2 3 4 5
1 2 3 4
1 2 3
1 2
1

Solution:

One possible solution.

Assembly Program	
	Addi x5,x0,#5
	Addi x1,x0,#1
	Add x2,x0,x5
	Add x3,x0,x0
AGAIN:	Add x3,x3,x1
	Out x3
	Bne x3,x2,**AGAIN**
	Out “\n”
	Add x3,x0,x0

	Sub x2,x2,x1
	Bne x2,x0,**AGAIN**
	Beq x0,x0,#0

Marking Scheme:

(a) If the number of loops are correct but the printed values are incorrect, award five(5) marks.
(b) If the pattern is correct but the number of iterations are incorrect, award five(5) marks.
(c) If the registers are not used to print the content. Some other means are used, award five(5) marks.
(d) If registers are used to print, the number of iterations are correct, and the pattern is correct, award ten(10) marks.

[CO4] Problem V : Cache Indexing **[03*3+01 Points]**

A processor having byte addressable range of 64GByte is connected to a 12-way set associative cache. The size of the block in cache is 64 Bytes and the cache size is 48KByte.

(c) Find the number of bits in tag, set index, and block offset
(d) For this cache system, if you were to place block number 64 (assume block numbers start from 0) from main memory to the cache. Which set number (assuming set number counting starts from 0) will you put it in?

Solution:

(a) Here,
Cache memory size = 48 KB
Block size = 64 B = 2^6 B
Main memory size = 64 GB = 2^36 B

The number of bits in the address = 36 bit
Number of bits in block offset = 6 bits [03 Points]

In a 12-way set-associative, each set has 12 blocks (or lines).
So, the total number of sets = (Cache size)/ (block size x Associativity)
= (48 KB)/(64 x 12)
= 64 = 2^6
The set index = 6 bits [03 Points]

The tag bit = number of bits in the address - (number of bits in set index + Number of bits on block)
= 36 - (6+6)
= 24 bit [03 Points]

(b) The block number 64 will be placed in 64 mod 64 i.e, set 0. [01 Point]

[CO4] Problem VI : Cache Miss Rate **[01+03+1 Points]**

Suppose you have a cache with a hit latency of 1 cycle. When it misses, it goes to main memory, which has a latency of 105 cycles (this includes the latency required to know that the access was a miss and the latency to bring the data from main memory). If the average memory access time (in cycles) is 6.2 cycles. What is the miss rate of the cache?

Solution:

Here,

Hit rate + miss rate =1 [01 Point]

Since

Average memory Access Time = hit_rate x hit latency + miss_rate x miss latency **[03 Points]**

=> 6.2 = (1- miss_rate) x 1 + miss_rate x 105

=> 6.2 = 1 - miss_rate + 105 x miss_rate

=> 5.2 = 104 x miss_rate

Miss rate = 5.2/104 = 0.05 = 5% [01 Point]

[CO1, CO3] Problem VII : Processor Speedup **[05 Points]**

You have built a non-pipelined processor with a frequency of 2.5 gigahertz, and the average cycles per instruction is 4. The same processor is upgraded to a pipelined processor with five stages, but due to the internal pipeline delay, the clock speed is reduced to 2 gigahertz. Assume there are no stalls in the pipeline. What is the ratio of speedup of the non-pipelined processor to the speedup of the pipeline processor for the execution of 100 instructions?

Solution:

Here,

For non-pipeline,

Time for single clock cycle = 1/2.5Ghz = 0.4 ns

Time taken for executing each instruction = 4 x 0.4 ns = 1.6ns

For 100 instructions = 1.6 x 100 ns = 160 ns [02 Points]

For pipeline,

Time taken for a single clock cycle = 1/2Ghz = 0.5 ns

Since there are no stalls in the pipeline,

If for exact 100 instructions:

Time taken to execute the hundred instructions = [5 + 100-1]x 0.5 = 52ns [02 Points]

Speedup = 160/52 = 3.0769 [01 Point]

If someone accounts for steady state:

Time taken to execute the hundred instructions = [100]x 0.5 = 50ns [02 Points]

Speedup = 160/50 = 3.2 [01 Point]

[CO2] Problem VIII : Caller-Callee **[10 Points]**

You need to convert the below-mentioned pseudo program into an assembly language program. The isa supports **First Block Instructions** as specified in the beginning of the question paper.

(h) The ISA has 32 general purpose registers x0,x1,....,x31
(i) x1 is the return address register
(j) x10, x11 stores the return value from the function.
(k) The arguments to the callee are stored in registers x12, x13, …, x17.
(l) The registers from x18, x19, …, x27 are caller saved.
(m) The registers x8, x9 are callee saved.
(n) Whenever the branch and link instruction is used. The return address is stored in x1, and the program counter jumps to the specified label.

Pseudo Program

```
Void func1()
{
        a=5
        b=9
        c=11
        d=func2(a,b,c)
        e=d+a
}
Int func2(a,b,c)
{
        return a+b-c
}
```

Solution

Note: Here we are not evaluating on the basis of saved register prospective, that which are caller saved and which are callee save. Otherwise we need to push and pop a lot of our register values onto the stack.

To feed the arguments to the function func2, we need to store them first in the argument registers i.e. x12,x13, … x17. And to return a value from the function **func2**, we need to use the return value register x10,x11.

Let's map the registers to the variables. x12(a), x13(b), x14(c), x15(d), x16(e).

Assembly Code

```
            Addi x12,x0,#5
            Add x8,x0,x12          // to save the content of a.
                                   // As, x8 is callee saved but x12 is not.
            Addi x13,x0,#9
            Addi x14,x0,#11
            Jal x1,FUNC2
            Add x15,x10,x0         // the caller is aware that res will be in x10.
                                   // As it is one of the return value register.
            Add x16,x15,x8
            Beq x0,x0,#0
FUNC2:      Add x10,x12,x13        // As x10 is the return value register.
            Add x10,x10,x14
```

Jalr x0,0(x1)

Marking Scheme:

(a) If the functionality is correct. And, If the registers used to pass the arguments to the subroutine **"FUNC2"** are chosen **correctly** i.e. among x12, … x17. But, the return value is **not stored** in the return value registers i.e. x10, x11. Here award five(5) marks. Here we are not evaluating the return address register.

(b) If the functionality is correct. And, if the registers used to pass the arguments to the subroutine **"FUNC2"** are chosen **incorrectly** i.e. among x12, … x17. But, the return value is **stored** in the return value registers i.e. x10, x11. Here award five(5) marks. Here we are not evaluating the return address register.

(c) If the functionality is correct. And, the registers for argument storing and returning are chosen correctly and the return value address is also used correctly, award ten(10) marks.

[CO3] <u>Problem IX: Branch Penalty</u> **[06 Points]**

Consider two processors C, D following RISC-V ISA and having stage description as follows..

(c) Processor C has 36 stages as F1, F2, . . F10, D1, D2, ... D9, E1, E2, . . . E9, M1, M2, . . M4, W1, W2, … W4. The branch operations are determined in the stage E6. The processor operates at 1GHz.

(d) Processor D has 92 stages as F1, F2, … F30, D1, D2, … D10, E1, E2, … E20, M1, M2, … M25, W1, W2, … W7. And, the branch operations are determined in the stage E6. The processor operates at 5GHz.

You need to calculate the Branch penalty of both the processors

(c) In terms of number of cycles.

(d) In terms of time lost in the penalty in ns.

<u>Solution</u>

Branch Penalty is the number of instructions to be squashed on a branch mispredict. In the processor microarchitecture that we are studying as in figure[2]. The Branch Penalty is the number instructions to be squashed which is equal to the number of pipeline stages before the stage that determines the branch. We have stage E8 in both the processors that determine the branch.

The branch penalty in Processor C is 10 + 9 + 5 = 24 cycles [02]
Time lost by Processor C during branch penalty = 24 * one_cycle_time
Time lost by Processor C during branch penalty = 24 * 1ns = 24ns [01]

The branch penalty in processor D is 30 + 10 + 5 = 45 cycles [02
Time lost by Processor C during branch penalty = 45 * one_cycle_time
Time lost by Processor C during branch penalty = 45 * 0.2ns = 9ns [01]