A) A processor embeds two cores that have private L1 and L2 caches, L3 is shared. The caches obey the MESI protocol and have the structure in the table: Addresses are 48-bit long. Write operations are managed with a write-back policy.

cache	dim. KB	block B	assoc.	latency clock	disp	fill-in index	fill-in tag
L1	128	32	2	4			
L2	2024	64	4	8			
L3	8192	64	8	16			

Assuming initially empty and invalidated cache lines throughout the hierarchy, consider the following program segment consisting of 3 instructions S1-S3:

S1 ADDI R1,R0,0000AAAAACC0hex
S2 LD F1,-8(R1)
S3 ST F1,8(R1)

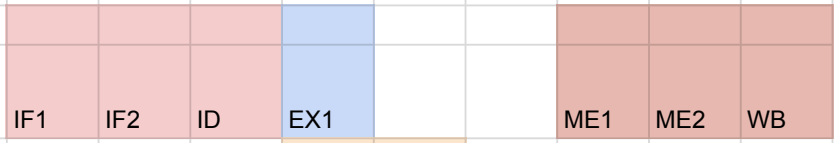
A1) Please draw the breakdown of the addresses involved in each instruction

		tag	index	disp
S1	L1			
	L2			
	L3			
S2	L1			
	L2			
	L3			
S3	L1			
	L2			
	L3			

A2) These instructions are executed in Core 1 and Core 2 with the following sequence
Fill in the bable specifying the MESI state of the blocks involved in all caches
and any action taken by the cache controllers

Sequence		Status of cache block in cache L of core C					Action taken by the cache controller
		L1 C1	L2 C1	L1 C2	L2 C2	L3	
Core 1	S1						
Core 1	S3						
Core 2	S1						
Core 2	S2						
Core 2	S3						
Core 1	S2						

Consider the statically scheduled pipeline drawn here aside: the pipeline has proper forwarding units that feed intermediate results to the ID/ stage; branch instructions take a decision in stage ME2.



Integer

Float add/sub

Float mult

B1 - prepare a producer/consumer table, properly choosing instruction classes

Consumer \ Producer						

	Prepare an optimized POE of code fragment below for the static pipeline of point B1			
	un-optimized			*optimized*
	`ADDI R1,R0,0000AAAAACC0hex`			
loop	`LD F3,0(R1)`			
	`ADDF F2,F1,F1`			
	`LD F4,-8(R1)`			
	`MULTF F4,F3,F2`			
	`ADDF F5,F4,F2`			
	`ST F5,8(R1)`			
	`ADDI R1,R1,16`			
	`BLI R1,0000AAAAF000hex,loop`			

	B3a) Show the ROE of the optimised POE (extend as necessary) the table		Clock cycle																
			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	B3b) Compute the CPI of the execution (showing proper values and formulas)																		

A) A processor embeds two cores that have private L1 and L2 caches, L3 is shared. The caches obey the MESI protocol and have the structure in the table: Addresses are 48-bit long. Write operations are managed with a write-back policy.

cache	dim. KB	block B	assoc.	latency clock	disp	index	tag
L1	128	32	2	4			
L2	2024	64	4	8			
L3	8096	64	8	16			

B4) Compute the number of D-cache hit and miss for the execution of the un-optimised loop, assuming the loop is executed on a single core on the cache hierarchy of exercise A) *repeated above*

```
        ADDI R1,R0,0000AAAAACC0hex
loop    LD F3,0(R1)
        ADDF F2,F1,F1
        LD F4,-8(R1)
        MULTF F4,F3,F2
        ADDF F5,F4,F2
        ST F5,8(R1)
        ADDI R1,R1,16
        BLI R1,0000AAAAF000hex,loop
```

C) Each core of the processor described in A) is organized as a superscalar, 2-way pipeline, that fetches, decodes issues and retires (commits) bundles containing each 2 instructions. The front-end in-order section (fetch and decode) consists of 3 stages for fetch and 2 stage for decode. The issue logic takes 1 clock cycle, if the instructions in the bundle are independent, otherwise it takes 2 clock cycles. The architecture supports dynamic speculative execution, and control dependencies from branches are solved when the branch evaluates the condition, even if it is not at commit. The execution model obeys the attached state transition diagram. There is a functional unit (FUs) Int1 for integer arithmetics (arithmetic and local instructions, branches and jumps, no multiplication), 1 FUs FAdd1 for floating point addition/subtraction, 1 FU FMolt1 for floating point multiplication, and a FU for division, FDiv1.

There are 8 integer (R0-R11) and 8 floating point (F0-F11) registers. Speculation is handled through a 8-entry ROB, a pool of 4 Reservation Stations (RS) Rs1-4 shared among all FUs, 1 load buffer Load1, 1 store buffer Store1 (see the attached execution model): an instruction bundle is first placed in the ROB (if two entries are available), then up to 2 instructions are dispatched to the shared RS (if available) when they are ready for execution and then executed in the proper FU. FUs are pipelined (except for the float division unit, which is blocking) and have the latencies quoted in the following table:

Int - 1	*Fadd – 2*
Fmolt – 2	*Fdiv – 8*

The processor embeds two cores that have private L1 and L2 caches, L3 is shared. The caches obey the MESI protocol and have the structure in the table: Addresses are 48-bit long. Write operations are managed with a write-back policy.

cache	dim. KB	block B	assoc.	latency clock
L1	128	32	2	4
L2	2024	64	4	8
L3	8096	64	8	16

The miss cost is 30 clock cycles

Show a schedule up to the issue of PC03 in the second iteration (extend as necessary the rows in the table)

PC	ISTRUCTION			INSTRUCTION STATE						
		n. ite	ROB pos	WO	RE	DI	EX	WB	RR	CO
PC01	ADDI R1,R0,0000AAAAACC0hex									
PC02	LD F3,0(R1)									
PC03	ADDF F2,F1,F1									
PC04	LD F4,-8(R1)									
PC05	MULTF F4,F3,F2									
PC06	ADDF F5,F4,F2									
PC07	ST F5,8(R1)									
PC08	ADDI R1,R1,16									
PC09	BLI R1,0000AAAAF000hex,PC02									

C2) Show the status at the time of issue of PC03 in the second iteration Time of issue:

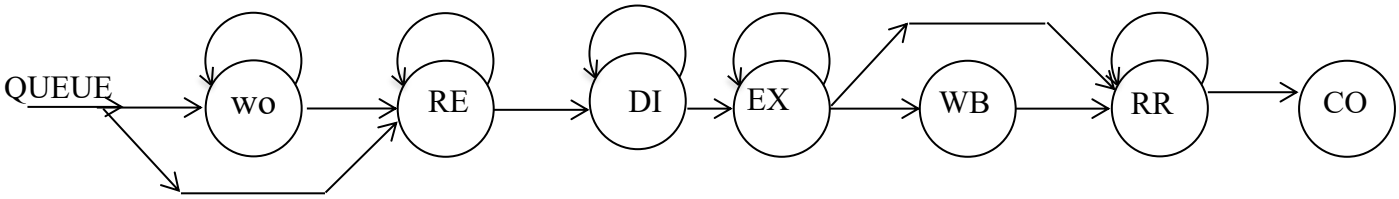
	Reservation station and load/store buffers							
	Busy	Op	Vj	Vk	ROBj	ROBk	ROB pos	Address
Rs1								
Rs2								
Rs3								
Rs4								
Load1								
Store1								

ROBj ROBk: sources not yet available

ROB pos: ROB entry number where instruction is located

	Result Register status							
Integer	R0	R1	R2	R3	R4	R5	R6	R7
ROB pos								
state								
Float.	F0	F1	F2	F3	F4	F5	F6	F7
ROB pos								
state								

Reorder Buffer (ROB)					
ROB Entry#	Busy	Op	Destination	Status	Value
0					
1					
2					
3					
4					
5					
6					
7					



Decoupled execution model for bundled (paired) instructions

The state diagram depicts the model for a dynamically scheduled, speculative execution microarchitecture equipped with a Reorder Buffer (ROB) and a set of Reservation Stations (RS). The ROB and RSs are allocated during the ISSUE phase, denoted as RAT (Register Alias Allocation Table) in INTEL microarchitectures, as follows: a bundle (2 instructions) if fetched from the QUEUE of decoded instructions and ISSUED if there is a pair of consecutive entries in the ROB (head and tail of the ROB queue do not match); a maximum of two instructions are moved into the RS (if available) when all of their operands are available. Access memory instructions are allocated in the ROB and then moved to a load/store buffer (if available) when operands (address and data, if proper) are available .

States are labelled as follows:

WO:	Waiting for Operands (at least one of the operands is not available)
RE:	Ready for Execution (all operands are available)
DI:	Dispatched (posted to a free RS or load/store buffer)
EX:	Execution (moved to a load/store buffer or to a matching and free UF)
WB:	Write Back (result is ready and is returned to the Rob by using in exclusive mode the Common Data Bus CDB)
RR:	Ready to Retire (result available or STORE has completed)
CO:	Commit (result is copied to the final ISA register)

State transitions happen at the following events:

from QUEUE *to* WO:	ROB entry available, operand missing
from QUEUE *to* RE:	ROB entry available, all operands available
loop at WO:	waiting for operand(s)
from WO *to* RE:	all operands available
loop at RE:	waiting for a free RS or load/store buffer
from RE *to* DI:	RS or load/store buffer available
loop on DI:	waiting for a free UF
from DI *to* EX:	UF available
loop at EX:	multi-cycle execution in a UF, or waiting for CDB
from EX *to* WB:	result written to the ROB with exclusive use of CDB
from EX *to* RR:	STORE completed, branch evaluted
loop at RR:	instruction completed, not at the head of the ROB, or bundled with a not RR instruction
from RR *to* CO:	bundle of RR instructions at the head of the ROB, no exception raised

Resources

Register-to-Register instructions hold resources as follows:

ROB: from state WO (or RE) up to CO, inclusive;

RS: state DI

UF: EX and WB

Load/Store instructions hold resources as follows:

ROB: from state WO (or RE) up to CO, inclusive;

Load buffer: from state WO (or RE) up to WB

Store buffer: from state WO (or RE) up to EX (do not use WB)

Forwarding: a write on the CDB (WB) makes the operand available to the consumer in the same clock cycle. If the consumer is doing a state transition from QUEUE to WO or RE, that operand is made available; if the consumer is in WO, it goes to RE in the same clock cycle of WB for the producer.

Branches: they compute Next-PC and the branch condition in EX and optionally forward Next-PC to the “in-order” section of the pipeline (Fetch states) in the next clock cycle. They do not enter WB and go to RR instead.