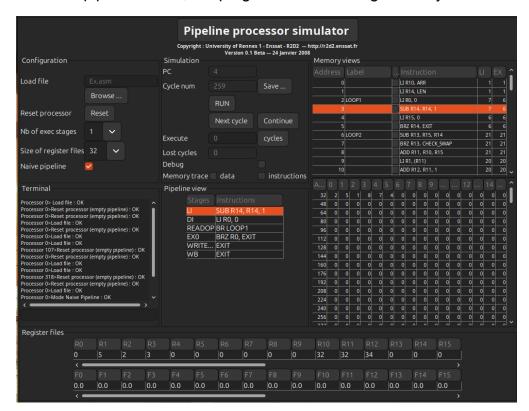
Nguyen Tai Anh- 22BI13028- Practical 2- Advanced CA and x86 ISA

I. Validate the correctness of program with and without naive mode

1. Without naive pipeline mode, the program runs normally as in Labwork 1:

				Р	ipel	ine p	roce	esso	r sin	nulat	or								
				Соруг	ight : Ur		Rennes 1			http://r2d2.	enssat.fr								
Configuration				Simulati	on	vers	ion U.1 Bet	ta 24 Ja	nvier 2008	Memor	y views								
				PC							s Lab		In					LI	EX ^
Load file				Cycle nur	n			Save			0			10, ARR					_1
	Вго	wse									1 2 LOOP	•		R14, LEN				6	
						RUN					3			3 R14, R14	4, 1			6	
Reset processor	Res	et				NeExec	ute the	progra	m from	PC until	pipelin	e empty	LI F	15, 0				5	5
Nb of exec stages											5		BR	Z R14, EXI					
no or executinges				Execute				cycles			6 LOOP	2		3 R13, R15 Z R13, CHI		/AD		20	20 20
Size of register file	s 32	~		Lost cycle	es						8			D R11, R10		iAr .		20	15
National address				Debug							9			R1, (R11)				20	
Naive pipeline				Memory	trace	data		instructions		10			AD	ADD R12, R11, 1				20	15 🗸
Toronto at						data		IIISCI	ictions	A 0	1 2	3 4	5 6	7 8		YY	12	. 14	
Terminal				Pipeline						32 1						0 0		0 0	
Processor 0> Load file : OK Processor 0> Reset processor (empty pipeline) : OK				tages				48 (0 (0 0	0 0	0 0		0 0	0	0 0	0		
Processor 0>Reset process	sor (empt					EXIT			64 (0 0		0 0		
Processor 0>Load file : OK Processor 0>Reset process		v ninelin	e) : OK		EADO					96 (0 0	0 0		0 0		0 0	
Processor 0>Load file : OK		y pipeiiii	.,. OK		X0	EXIT				112			0 0	0 0		0 0	0	0 0	
Processor 0>Load file : OK Processor 107>Reset processor 107		nnty nine	ine) · OK		VRITE					128 (0 0			
Processor 0>Reset process	sor (empt			<u>v</u>	٧B	NOP (S	SUB R14	, R14		144 (0 0	0 0		0 0	0	0 0	
Processor 0>Load file : OK Processor 0>Load file : OK										160 (0 0		0 0	
Processor 318>Reset proc	essor (en									176 (192 (0 0	0 0		0 0	0	0 0	
Processor 0>Reset process Processor 0>Load file : OK		y pipelin	e) : OK							208 (0 0	0 0		0 0	0	0 0	
Processor 0>Mode Naive		OK								224 (0 0		0 0	
~	_	_	- >							240 (0 0	0 0		0 0		0 0		0 0	
										256 (0 0				0 0		0 0	0 0
Register files										7771 (AI A	
RC) R	1	R2 R3	R4	R5	R6	R7	R8	R9	R10	R11	R12	R13	R14	R1				
0				0	0	0	0	0	0	32	32	33	0						
· · · · · ·								•											
FO) F	1	F2 F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F1				
0.0	0 0	.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0)			
<u> </u>	_							-											

2. With naive pipeline mode, the program is not running correctly:

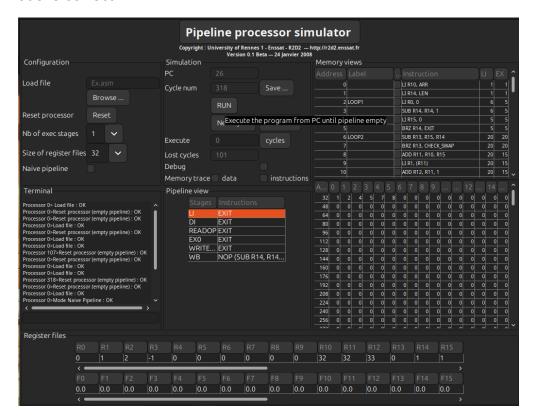


3. Explanation:

- When executing in normal pipeline mode, results become more accurate because each instruction is executed in distinct stages of the pipeline. This ensures that the outcome of each instruction is correctly processed before continuing to the next stage. The value from the previous instruction must reach the Write Back (WB) stage and be updated before the next dependent instruction is executed. For example, "add R13, R14, R15", the result of R12 = R14 + R15 will be saved first and then move to the next instruction such as Ii R0, R13.
- The reason why naive pipeline mode will run incorrectly is because in naive pipeline mode, instructions dependent on the results of previous instructions may not wait for those results. For example, if R14 and R15 are loaded from memory and immediately compared or swapped/add/subtract, the compare or swap might execute before the load completes or before moving to the next instruction related to these parameters, leading to the fact that the result will be failed and making the execution incorrect.

II. Analyze the execution of bubble algorithm in pipeline:

 Note if the execution is correct or not: The execution without naive pipeline mode is correct.



2. Capture case when two instructions are dependents and understand what happens

sub R13, R15, R14 brz R13, .CHECK R13 = R15 - R14, and loop until R13 = 0, continue at CHECK Instruction.

add R11, R10, R15 li R1, (R11)

R11 = R10 - R15, after that the value of R11 will be load to R1

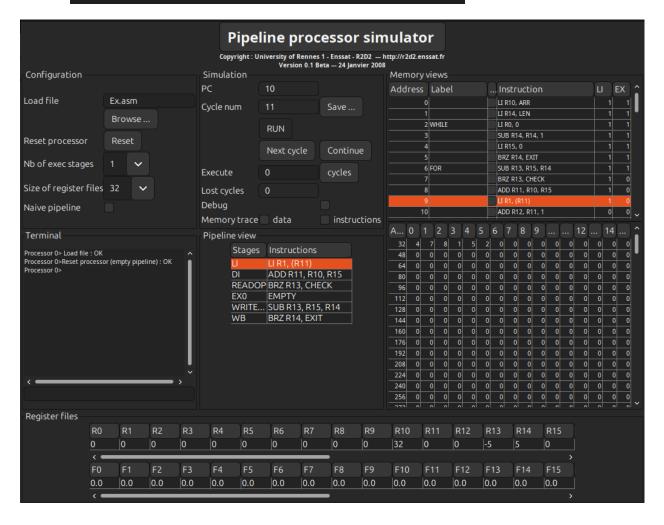
add R12, R11, 1 li R2, (R12)

R12 = R11 - R1, after that the value of R12 will be load to R2

3. Capture case when a branch is executed and understand what happens

When a branch is executed, the instructions inside the branch will be executed in the new flow, and the instructions after the branch will not be executed until the branch has been resolved.

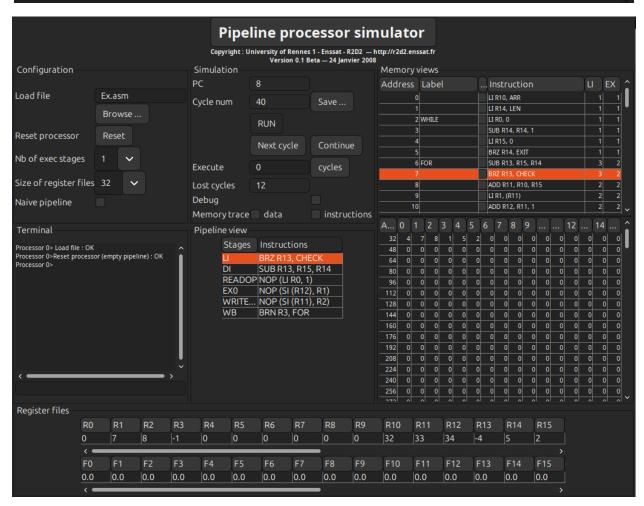
brz R14, EXIT -- if length is zero, exit



In this case, firstly, I already have instructions to calculate the length of the initialize array to implement the loop, then, I set this value to R14 and decrease by 1 when one loop is finished. After this value of R14 become 0, the loop will end and exit the program to avoid the infinite loop

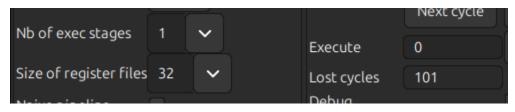
Another example is:





In this case, the value of R3 is R1 - R2 which R1 is the value of one element in the array and R2 is the next element of R1 in the array. If R3 is negative (brn R3), that means that R1 < R2 \rightarrow R1 and R2 will not swap due to the bubble sort algorithm.

4. Note the number of cycles which are lost during the execution with different number of pipeline stages



NIL of average and			Next cycle
Nb of exec stages	2	Execute	0
Size of register files	32 🗸	Lost cycles	204
Nietus-te-ti		Debug	Next cycle
Nb of exec stages	3	Execute	0
Size of register files	32 🗸	Lost cycles	296
Naive eigeliee		Debug	
Nb of exec stages	4 🗸		Next cycle
140 of exec stages		Execute	0
Size of register files	32	Lost cycles	388
National aller		Debug	Next cycle
Nb of exec stages	5 🗸	Execute	0
Size of register files	32	Lost cycles	347
National adias		Debug	341
			Next cycle
Nb of exec stages	6	Execute	0 Execute on
Size of register files	32	Lost cycles	418
Mainerialia		Debug	
Nb of exec stages	7 🗸		Next cycle
ND OF EXEC stages	بنيك	Execute	0
Size of register files	32 🗸	Lost cycles	489
National alles		Debug	Next cycle
Nb of exec stages	8 🗸		
}	22	Execute	0
Size of register files	32	Lost cycles	560
Maiora di alian		Debug	

	Ţ			Next cycle
Nb of exec stages	9	*	Execute	0
Size of register files	32	~	Lost cycles	631
Matina de altera	=		Debug	
	Y			Next cycle
Nb of exec stages	10	~		
Nb of exec stages	10	<u>~</u>	Execute	Next cycle
Nb of exec stages Size of register files		*	Execute Lost cycles	

→ **Explanation:** When the instruction is executed in different stages, there are different numbers of instructions executed at the same time, so that there will be different amounts of cycles lost during the execution normally, the bigger the number of stages, the bigger the number of lost cycles. When the number of execution stages is increased, each instruction must pass through more stages before it is completed, which might lead to a higher number of cycles in the pipeline (in some cases, this number of cycles might reduce rather than increase). Even though the total number of cycles and potential lost cycles may rise, the crucial point is that the final output remains correct. Adding more execution stages boosts pipeline performance by enabling concurrent processing of multiple instructions and increasing parallelism.