VLSI Architecture Design Course (048853)
Final Exam
July 14th, 2004

Electrical engineering Department

Student name: ______________________ Student number:________________

This exam contains TWO questions.
The exam duration is 2:30 hours.
Please fill the answers ON THE EXAM forms.

Please explain or provide a formula for each computation!

TAKE YOUR TIME, READ THE QUESTIONS THOUROUGHLY, UNDERSTAND THE CONTENT AND ONLY THEN START TO ANSWER

Good luck!

Q1	
Q2	
Total	

Question 1 (50%) Cache Power Management

Nowadays, cache power consumption mainly consists of leakage power, caused by relatively low activity for many transistors. Several mechanisms exist that reduce the leakage (static) power.

We define the *Live Time* of a block (cache line) as the time between its allocation until its last access before eviction.

We define the Dead Time of a block (cache line) as the time between its last access until the block is evicted from the cache.

These definitions are explained in Figure 1.1:

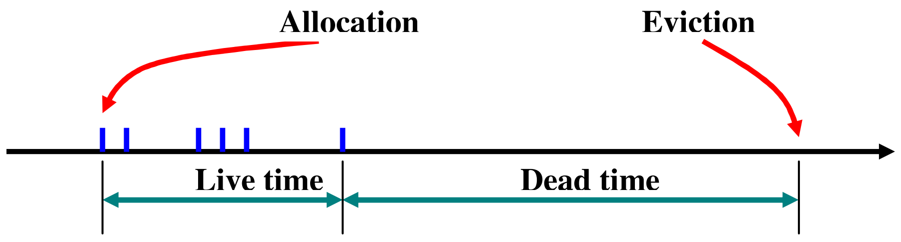

Figure 1.1

We define the *dead time ratio* as the dead time divided by the sum of the dead time and the live time of the block.

Cache blocks are usually dead rather than alive, as can be seen in Figure 1.2.

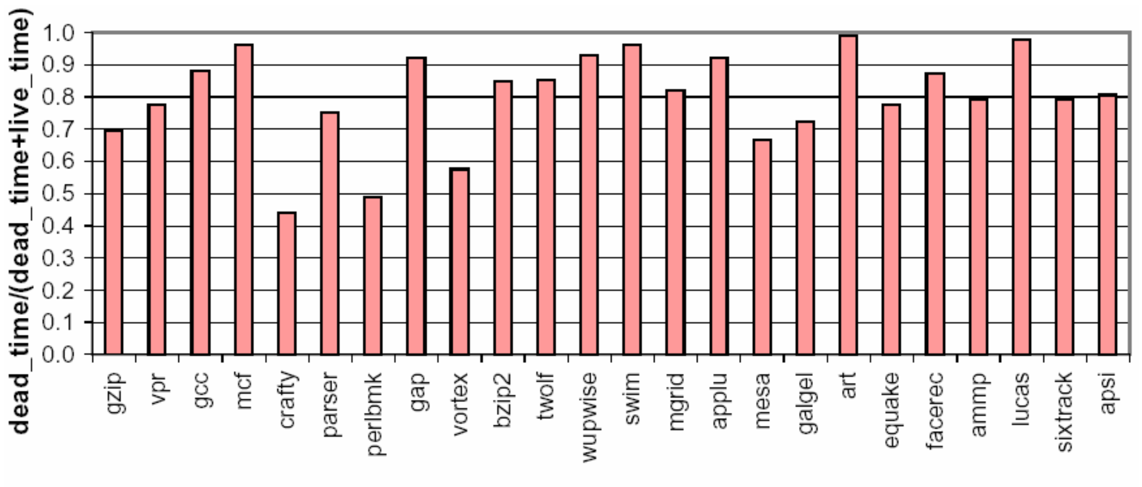

Figure 1.2 – Dead Time Ratio for various benchmarks

*This question is based on the following:
[1] S. Kaxiras et al., "Cache Decay: Exploiting Generational Behavior to Reduce Cache Leakage Power", Proc. ISCA 2001
[2] M. Powell et al., "Gated Vdd: A Circuit Technique to Reduce Leakage in Deep-Submicron Cache Memories", Proc. ISLPED 2000
[3] K. Fkautner et al., "Drowsy Caches: Simple Techniques for Reducing Leakage Power", Proc. ISCA 2002

QUESTIONS:

A. How does the miss rate affect the dead time ratio? Explain.

B. Describe how each of the following parameters affects the dead time ratio. Use the following table as a supporting tool.

	Miss rate (+=increase)	Replacement rate (+=increase)	Dead time ratio (+=increase)
Increasing cache size			
Increasing block size on the expense of the number of sets			

Explanations:

1. Increasing cache size

2. **Increasing cache block size on the expense of the number of sets**

C. In order to conserve power, "dead" cache blocks (cache lines) should be cut off from power (i.e. content is erased). This method is called "Cache Decay". An example of one bit of decay cache controlled by the gated-VDD control signal is shown in figure 1.3.

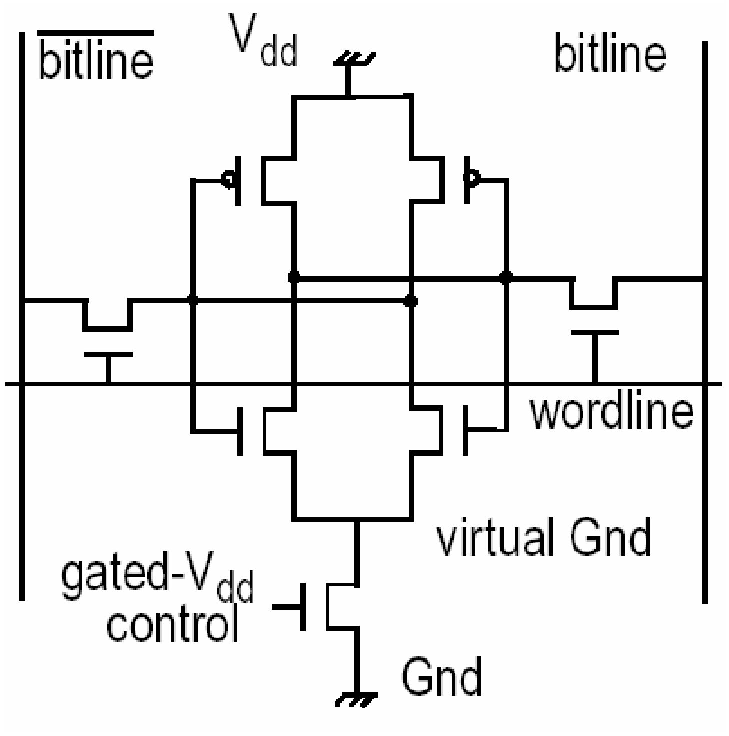

Figure 1.3 – Decay cache bit circuit

One of the ways to detect when a cache block is "dead" is by counting the number of cycles that have passed since the last access to that block.

1. **How will you decide when to cut off a block from power?**

2. **What are the disadvantages of using this method?**

3. **What happens in case a block is not really "dead"? What are the implications?**

4. **What can the software developer do in order to avoid eviction of live blocks?**

5. **Can you think of a way to reduce the cost of counting?**

D. "Drowsy Cache" is a method that puts suspected dead blocks to sleep by lowering their supply voltage. When drowsed blocks are accessed, the cache needs to first wake up the drowsed blocks by increasing their supply voltage. This wake up process takes a few cycles. The number of cycles is called the wake-up penalty. An example of a drowsy line is shown in Figure 1.3.

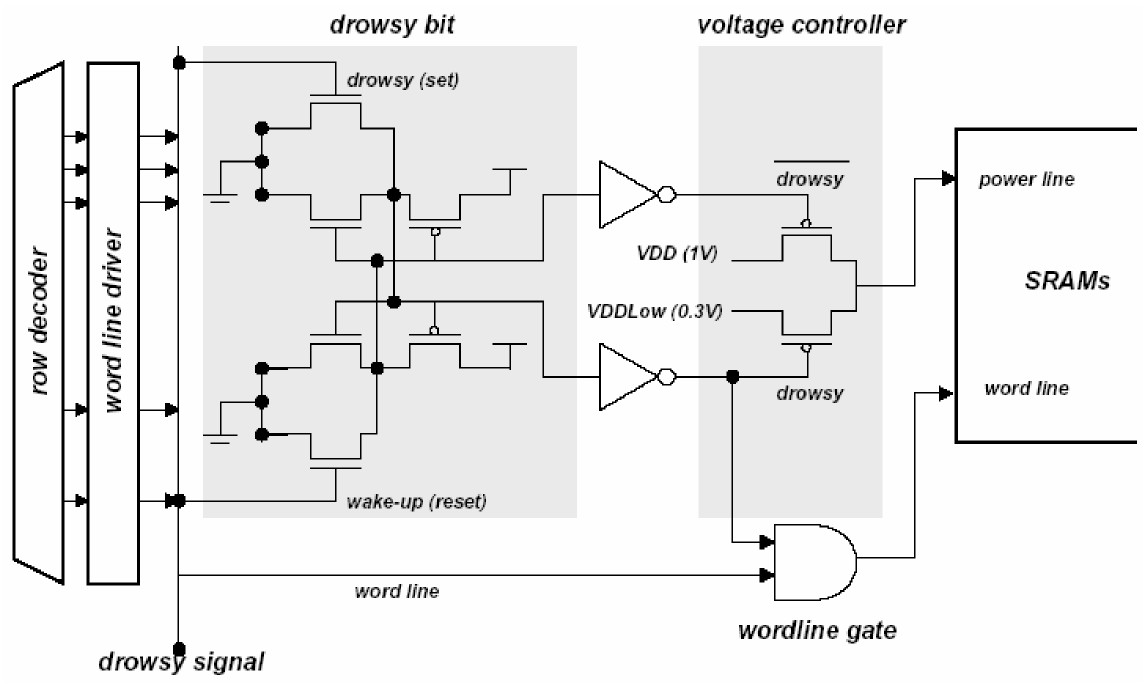

Figure 1.4 – Drowsy cache line circuit

1. **What are the advantages of using Drowsy Cache over Cache Decay?**

2. **What are the disadvantages of using Drowsy Cache over Cache Decay?**

3. **Can you think of a way to use a drowsy cache without using a counter for each line?**

Question 2 (50%) Processor Core Performance

In this question deals with the execution of instructions within a processor core accounting for simple cache and branch prediction effects.

Assume the following 2 processor core configurations:

	In-order core	Out-of-order core
Fetch bandwidth	16 consecutive instructions Restart a new fetch in the following cases: Correctly predicted taken branch: 1 cycle latency All mispredicted branch: after the branch execution stage	
Execution bandwidth	Unlimited	
L1 Instruction cache	Perfect (100% hit)	
L1 data cache	32KB, 2 Ways, 64 Bytes lines, writeback cache	
branch predictor	Simple local 2 bit counter (bi-modal) Static prediction: Forward: strongly not taken, Backward: strongly taken	
Pipeline depth	4 (F/D/E/W) Fetch/Decode/ Execute/Writeback	6 (F/D/R/S/E/W) Fetch/Decode/Rename/ Schedule/Execute/Writeback
ALU instruction execution latency	1 cycle	
Load Instruction execution latency	L1 hit: 3 cycles, L1 miss: 10 cycles 16 outstanding cache/memory accesses can co-exist A memory element that is within a line that is being brought into the cache is available one cycle after the missed data item is available.	

We will examine the performance of our cores on the following 2 code fragments (A and B).
The programs sum a subset of array element based on some criteria.
"int" is 4 bytes (32 bit element) in this code.

	(A) Simple "sum"	(B) Conditional "sum"

C code	int a[N], i, sum;
	sum = 0;
	for (i=0; i<N; i++)
	    sum += a[i];
		int a[N], i, sum;
		sum = 0;
		for (i=0; i<N; i++)
		    if (a[i]>0)
		        sum += a[i];

Assembly code	mov 0, Rsum
	     mov 0, Ri
	L1: load a[Ri], R1
	
	     add R1, Rsum
	     add 1, Ri, Ri
	     cmp Ri, N
	     jle L1
	END: …
		mov 0, Rsum
		     mov 0, Ri
		L1: load a[Ri], R1
		     cmp R1, 0
		     jle L2
		     add R1, Rsum
		L2: add 1, Ri, Ri
		     cmp Ri, N
		     jle L1
		END: …

QUESTIONS:

Assume that:

I. **L1 cache is empty before we start.**
II. **N = 1,000,000**
III. **For simplicity, ignore the 1st two assembly instructions (sum=0; i=0).**
IV. **For simplicity, assume that there is an infinite instruction window.**
V. **You can ignore edge effects**
it is OK if an answer is within 0.1% of the accurate result.

A. Let us look at code portion (A).
We follow its execution until it reaches the label END.

1. **How many cache misses are in this piece of code? Explain.**

2. **How many branch mispredictions are in this piece of code? Explain.**

3. **For the in-order core configuration: (The solution is provided for your convenience, please see also the instruction/cycle chart marked A.I that appears later.)**
   How many cycles will it take to execute? __*5437500*__
   What is the IPC? __*0.92*__
   Explain.
   Account for dependencies, cache misses and branch mispredictions.

4. **Repeat for the <u>Out-of-order</u> core configuration:**
   How many cycles will it take to execute? ___________
   What is the IPC? __________
   Explain.
   Account for dependencies, cache misses and branch mispredictions.
   For your convenience, you many use the instruction/cycle chart marked A.O that appears later.

B. **We now perform a similar exercise on the code portion (B). Assume:**

I. **No cache misses.**
II. **The array "a" is set so**
   For all i, a[i]=1, except every 5th element starting from i=1, then it is -1.
   Or more formally: a[i]=-1 if i is of the form 5*k+1 (1, 6, 11…).

$$a[i]=\begin{Bmatrix}1 & i\neq 5k+1\\ -1 & i=5k+1\end{Bmatrix}$$

1. **How many branches are taken in this code fragment? Explain.**
   JLE L1: Taken _____________ , Not Taken _________________
   JLE L2: Taken _____________ , Not Taken _________________

2. **How many instructions were committed (retired)? ______________**
   Explain.

3. How many branches were mispredicted in this piece of code? Explain.

   JLE L1: Predicted Not Taken, actually Taken ________________
   Predicted Taken, actually Not Taken ________________

   JLE L2: Predicted Not Taken, actually Taken ________________
   Predicted Taken, actually Not Taken ________________

4. For the in-order core configuration:
   How many cycles it will take to execute? ___________
   What is the IPC? __________
   Explain.
   Account for dependencies and branch mispredictions.
   For your convenience, you many use the instruction/cycle chart marked A.I that appears later.

5. Repeat for the Out-of-order core configuration:
   How many cycles it will take to execute? ___________
   What is the IPC? __________
   Explain.
   Account for dependencies and branch mispredictions.
   For your convenience, you many use the instruction/cycle chart marked A.I that appears later.

C. What have you learnt from this example about the difference between in-order and out-of-order potential.

1. **Exploiting instruction level parallelism.**

2. **Sensitivity to cache misses.**

3. **Sensitivity to the quality of the branch predictor.**

D. **It was suggested to speed up this code by using an out-of-order Simultaneous Multi Threading (SMT) machine.**

The code is split into 2 threads, implemented by 2 distinct functions (f1 and f2). The first thread executes the function f1 that performs the first 500,000 iterations. The second thread executes the function f2 that performs the last 500,000 iterations.

On each cycle the processor can fetch instructions from one thread only (up to 16 consecutive instructions). The processor can decide from which thread it can fetch next.

1. **What is the advantage of using SMT in these cases:**

   For code portion A?

   For code portion B?

2. **For code portion B, what will be an ideal fetch policy (from which thread to fetch next) for this code, so to maximize performance?**

3. **For code portion B, estimate the performance benefit in such case.**

A.I: Sheet for code portion (A) in-order

		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60
1	load a[Ri], R1	F	D	E	E	E	E	E	E	E	E	E	E	W																																															
2	add R1, Rsum	F	D											E	W																																														
3	add 1, Ri, Ri	F	D											E	W																																														
4	cmp Ri, N	F	D												E	W																																													
5	jle L1	F	D													E	W																																												
6	load a[Ri], R1		F	D												E	E	E	W																																										
7	add R1, Rsum		F	D															E	W																																									
8	add 1, Ri, Ri		F	D															E	W																																									
9	cmp Ri, N		F	D																E	W																																								
10	jle L1		F	D																	E	W																																							
11	load a[Ri], R1			F	D																E	E	E	W																																					
12	add R1, Rsum			F	D																			E	W																																				
13	add 1, Ri, Ri			F	D																			E	W																																				
14	cmp Ri, N			F	D																				E	W																																			
15	jle L1			F	D																					E	W																																		
16	load a[Ri], R1				F	D																				E	E	E	W																																
17	add R1, Rsum				F	D																							E	W																															
18	add 1, Ri, Ri				F	D																							E	W																															
19	cmp Ri, N				F	D																								E	W																														
20	jle L1				F	D																									E	W																													
21	load a[Ri], R1																																																												
22	add R1, Rsum																																																												
23	add 1, Ri, Ri																																																												
24	cmp Ri, N																																																												
25	jle L1																																																												
26	load a[Ri], R1																																																												
27	add R1, Rsum																																																												
28	add 1, Ri, Ri																																																												
29	cmp Ri, N																																																												
30	jle L1																																																												
31	load a[Ri], R1																																																												
32	add R1, Rsum																																																												
33	add 1, Ri, Ri																																																												
34	cmp Ri, N																																																												
35	jle L1																																																												
36	load a[Ri], R1																																																												
37	add R1, Rsum																																																												
38	add 1, Ri, Ri																																																												
39	cmp Ri, N																																																												
40	jle L1																																																												
41																																																													

A.O: Sheet for code portion (A) out-of—order

		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60
1	load a[Ri], R1	F	D	R	S	E	E	E	E	E	E	E	E	E	E	W																																													
2	add R1, Rsum	F	D	R	S											E	W																																												
3	add 1, Ri, Ri	F	D	R	S	E											W																																												
4	cmp Ri, N	F	D	R	S		E										W																																												
5	jle L1	F	D	R	S			E									W																																												
6	load a[Ri], R1		F	D	R	S	E	E	x	x	x	x	x	x	x	E	W																																												
7	add R1, Rsum		F	D	R	S											E	W																																											
8	add 1, Ri, Ri		F	D	R	S	E											W																																											
9	cmp Ri, N		F	D	R	S		E										W																																											
10	jle L1		F	D	R	S			E									W																																											
11	load a[Ri], R1																																																												
12	add R1, Rsum																																																												
13	add 1, Ri, Ri																																																												
14	cmp Ri, N																																																												
15	jle L1																																																												
16	load a[Ri], R1																																																												
17	add R1, Rsum																																																												
18	add 1, Ri, Ri																																																												
19	cmp Ri, N																																																												
20	jle L1																																																												
21	load a[Ri], R1																																																												
22	add R1, Rsum																																																												
23	add 1, Ri, Ri																																																												
24	cmp Ri, N																																																												
25	jle L1																																																												
26	load a[Ri], R1																																																												
27	add R1, Rsum																																																												
28	add 1, Ri, Ri																																																												
29	cmp Ri, N																																																												
30	jle L1																																																												
31	load a[Ri], R1																																																												
32	add R1, Rsum																																																												
33	add 1, Ri, Ri																																																												
34	cmp Ri, N																																																												
35	jle L1																																																												
36	load a[Ri], R1																																																												
37	add R1, Rsum																																																												
38	add 1, Ri, Ri																																																												
39	cmp Ri, N																																																												
40	jle L1																																																												
41																																																														

B.I: Sheet for code portion (B) – In-order

			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60
1	load	a[Ri], R1	F	D	E	E	E	W																																																						
2	cmp	R1, 0	F	D				E	W																																																			
3	jle	L2	F	D					E	W																																																	
4	add	R1, Rsum	F	D					E	W																																																	
5	add	1, Ri, Ri	F	D					E	W																																																	
6	cmp	Ri, N	F	D						E	W																																															
7	jle	L1	F	D							E	W																																														
8	load	a[Ri], R1		F	D						E	E	E	W																																												
9	cmp	R1, 0		F	D									E	W																																											
10	jle	L2		F	D										E	W																																										
11	add	R1, Rsum		-	-	-	-	-	-	-	-	-	-	-	-	-																																										
12	add	1, Ri, Ri		-	-	-	-	-	-	-	-	-	-	-	-	F	D	E	W																																						
13	cmp	Ri, N		-	-	-	-	-	-	-	-	-	-	-	-	F	D		E	W																																					
14	jle	L1		-	-	-	-	-	-	-	-	-	-	-	-	F	D			E	W																																				
15	load	a[Ri], R1															F	D		E	E	E	W																																						
16	cmp	R1, 0																																																										
17	jle	L2																																																										
18	add	R1, Rsum																																																										
19	add	1, Ri, Ri																																																										
20	cmp	Ri, N																																																										
21	jle	L1																																																										
22	load	a[Ri], R1																																																										
23	cmp	R1, 0																																																										
24	jle	L2																																																										
25	add	R1, Rsum																																																										
26	add	1, Ri, Ri																																																										
27	cmp	Ri, N																																																										
28	jle	L1																																																										
29	load	a[Ri], R1																																																										
30	cmp	R1, 0																																																										
31	jle	L2																																																										
32	add	R1, Rsum																																																										
33	add	1, Ri, Ri																																																										
34	cmp	Ri, N																																																										
35	jle	L1																																																										
36	load	a[Ri], R1																																																										
37	cmp	R1, 0																																																										
38	jle	L2																																																										
39	add	R1, Rsum																																																										
40	add	1, Ri, Ri																																																										
41	cmp	Ri, N																																																										
42	jle	L1																																																										
43																																																												

elements marked with "-" denote flushed operations.

B.O: Sheet for code portion (B) – Out-of-order

		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60
1	load a[Ri], R1	F	D	R	S	E	E	E	W																																																			
2	cmp R1, 0	F	D	R	S				E	W																																																		
3	jle L2	F	D	R	S					E	W																																																
4	add R1, Rsum	F	D	R	S				E		W																																																
5	add 1, Ri, Ri	F	D	R	S	E					W																																																
6	cmp Ri, N	F	D	R	S		E				W																																																
7	jle L1	F	D	R	S			E			W																																																
8	load a[Ri], R1		F	D	R	S	E	E	E		W																																																
9	cmp R1, 0		F	D	R	S				E	W																																																
10	jle L2		F	D	R	S					E	W																																														
11	add R1, Rsum		-	-	-	-	-	-	-	-	-	-	-	-	-	-	-																																								
12	add 1, Ri, Ri		-	-	-	-	-	-	-	-	-	F	D	R	S	E	W																																								
13	cmp Ri, N		-	-	-	-	-	-	-	-	-	F	D	R	S		E	W																																							
14	jle L1		-	-	-	-	-	-	-	-	-	F	D	R	S			E	W																																						
15	load a[Ri], R1												F	D	R	S	E	E	E	W																																					
16	cmp R1, 0																																																												
17	jle L2																																																												
18	add R1, Rsum																																																												
19	add 1, Ri, Ri																																																												
20	cmp Ri, N																																																												
21	jle L1																																																												
22	load a[Ri], R1																																																												
23	cmp R1, 0																																																												
24	jle L2																																																												
25	add R1, Rsum																																																												
26	add 1, Ri, Ri																																																												
27	cmp Ri, N																																																												
28	jle L1																																																												
29	load a[Ri], R1																																																												
30	cmp R1, 0																																																												
31	jle L2																																																												
32	add R1, Rsum																																																												
33	add 1, Ri, Ri																																																												
34	cmp Ri, N																																																												
35	jle L1																																																												
36	load a[Ri], R1																																																												
37	cmp R1, 0																																																												
38	jle L2																																																												
39	add R1, Rsum																																																												
40	add 1, Ri, Ri																																																												
41	cmp Ri, N																																																												
42	jle L1																																																												
43																																																													

elements marked with "-" denote flushed operations.