NATIONAL UNIVERSITY OF SINGAPORE

ANSWERS

CS2100 – COMPUTER ORGANISATION

(Semester 2: AY2017/18)

ANSWER BOOKLET

Time Allowed: 2 Hours

INSTRUCTIONS TO CANDIDATES

1. This answer booklet consists of **SIX (6)** printed pages.
2. Fill in your Student Number **with a pen clearly** below. Do NOT write your name.
3. You may write your answers in pencil (2B or above).

STUDENT NUMBER (fill in with a pen):

A	0	1						

For examiner's use only		
Question	***Total***	***Marks***
Q1	10	
Q2	15	
Q3	20	
Q4	12	
Q5	15	
Q6	14	
Q7	14	
Total	**100**	

Write your answers in the box/space provided.

1a. [4]

```
11 A
33 f
```

1b. [3]

123.125

42F64000 = 0 <u>100 0010 1</u>111 0110 0100 0000 0000 0000

10000101 = 133 – 127 = 6

$1.111011001_2 \times 2^6 = 1111011.001_2 = 123.125$

1c. [3]

F = **1**

Circuit delay = ***9t***

Q1: /10

2a. [10]

DA = $\boldsymbol{A \cdot C' + B \cdot C' \cdot D'}$

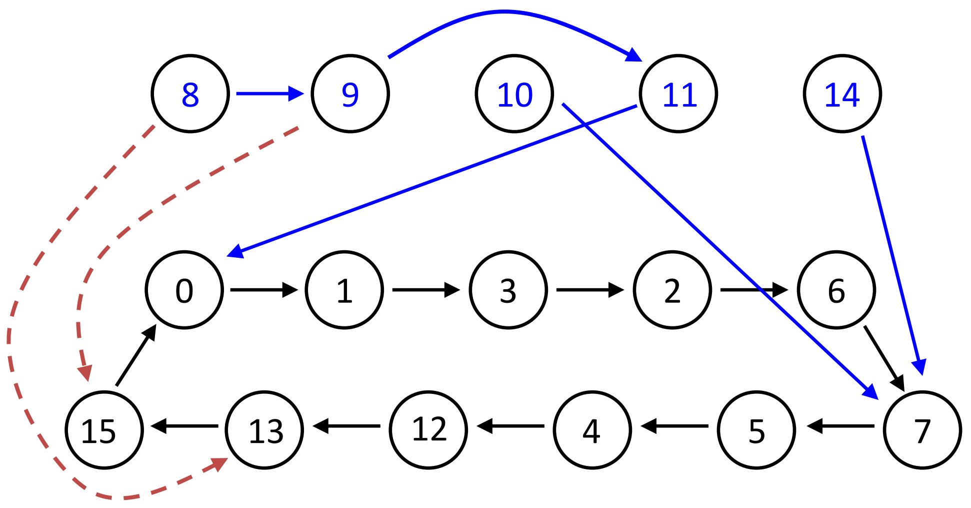
DB = $\boldsymbol{A' \cdot B + C \cdot D' + B \cdot C'}$ or $\boldsymbol{A' \cdot B + C \cdot D' + A \cdot C'}$

TC = $\boldsymbol{A \cdot D + B \cdot C \cdot D + B' \cdot C' \cdot D}$

JD = $\boldsymbol{A + B \cdot C + B' \cdot C'}$

KD = $\boldsymbol{A \cdot C + B' \cdot C + A' \cdot B \cdot C'}$

2b. [5]



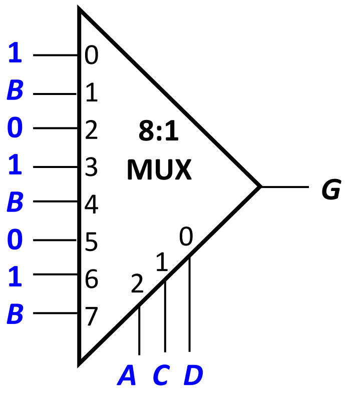
There are two sets of answers, where states 8 and 9 may transit to different states.
Set 1: 8 → 9; 9 → 11 (Correspond to $DB = A' \cdot B + C \cdot D' + B \cdot C'$)
Set 2: 8 → 13; 9 → 15 (Correspond to $DB = A' \cdot B + C \cdot D' + A \cdot C'$)

Q2: /15

3a. [4]

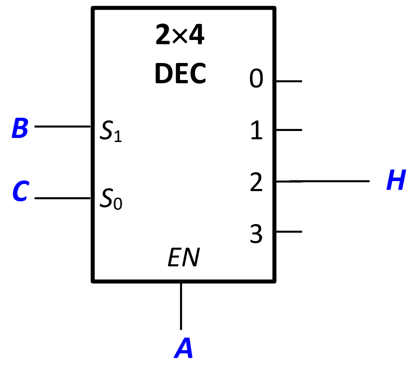
$F =$ **A**

3b. [4]



3c. [4]

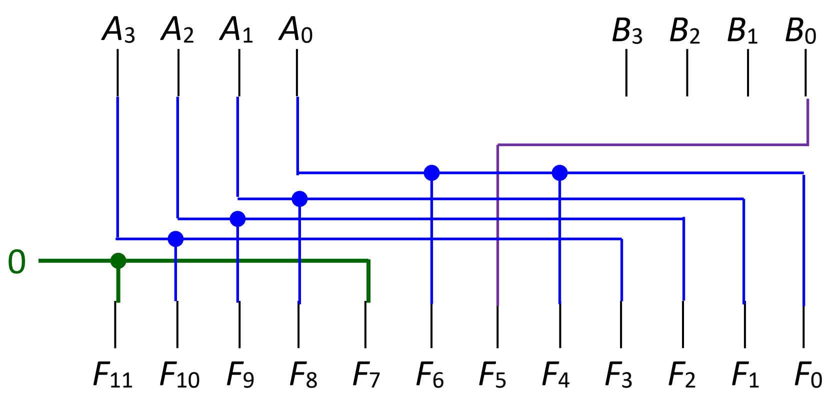
Many alternative answers.



3d. [8]

A				$5 \times A$							
A_3	A_2	A_1	A_0								
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	1	0	1
0	0	1	0	0	0	0	1	0	0	0	0
0	0	1	1	0	0	0	1	0	1	0	1
0	1	0	0	0	0	1	0	0	0	0	0
0	1	0	1	0	0	1	0	0	1	0	1
0	1	1	0	0	0	1	1	0	0	0	0
0	1	1	1	0	0	1	1	0	1	0	1
1	0	0	0	0	1	0	0	0	0	0	0
1	0	0	1	0	1	0	0	0	1	0	1

Complete your circuit here.



Q3: /20

4a.
[2]

Maximum total instructions = **379**

$(2^6 - 5) + (5 \times 2^6) = 59 + (5 \times 64) = 59 + 320 = 379$

4b.
[3]

Stuck-at-0 fault at bit 6 of the instruction

Eg: **addi $t0, $zero, 64**

This is supposed to put the value 64 into $t0. However, due to the stuck-at-0 error at bit 6, the value in $t0 will be 0.

Other similar examples using the same argument are acceptable, for example, some students use **sll $t1, $t1, 1** (pre-condition: $t1 cannot be zero), but this would require comparing the old and new values of $t1. The **addi** solution above is simpler.

4c.
[3]

Stuck-at-0 fault at ALUSrc

If the instruction is correctly carried out, **lw $t1, 0($t0)** would have loaded the value at address 12 into $t1.

With stuck-at-0 fault at ALUSrc, the instruction would have loaded the value at address 46 into $t1 instead.

Hence, we could first load different values in addresses 12 and 46 before using the test.

4d.
[4]

Adding bne instruction

Since ALUop code 11 is not used, we may use it for bne.
Hence, branch taken = **ALUop1** AND **ALUop2** AND **!(isZero)**
where (isZero) is the output from the ALU.

or

Use ALUop code 01 (same as beq).
Hence, branch taken = !**ALUop1** AND **ALUop2** AND **!(isZero)**
where (isZero) is the output from the ALU, or
branch taken = **Branch** AND **!(isZero).**

I tried to be more liberal in marking this part as most students didn't give a sufficiently rigorous answer.

Q4: /12

5a. [1]

lw $s2, 0($t0)

5b. [4] Array *A*:

11	10	31	14	9	42	6	11

These elements are unchanged.

5c. [4]

```
int i;
for (i=n-1; i>=0; i-=2) {
  if (B[i]%4 == 3)
     A[i]++;
  else
     A[i] += B[i];
}
```

5d. [2] 0018c080

5e. [2] 0810001c

5f. [2] 0300782a

Q5: /15

6a. [2]

Minimum = 1203 (3 + 100×12)

Maximum = 1303 (3 + 100×13)

6b. [6]

(Due to control) Inst2, Inst4, Inst11, Inst13, Inst14
(Due to data) Inst8, Inst10, Inst17

6c. [3] 24 cycles

6d. [3] 26 cycles

Q6: /14

7a.
[2]

Index: __5__ bits; Offset: __4__ bits

7b.
[4]

A[1023] → Index __31__; *B*[1023] → Index __15__;

7c.
[2]

Array *A*: __1024__ accesses; Array *B*: __512__ accesses

7d.
[2]

Array *A*: __75__ %; Array *B*: __50__ %

7e.
[4]

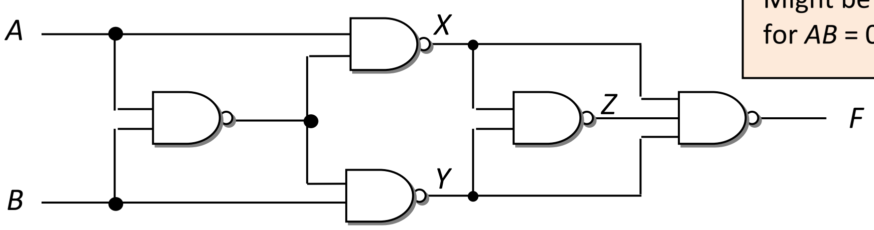
Misses: __9__; Hits: __6138__

Q7: /14

=== END OF PAPER ===

Workings

Q1c.



Might be easier just to trace for AB = 00, 01, 10, 11.

$X = ((A\cdot B)'\cdot A)' = A\cdot B + A' = B + A'$; $\quad Y = ((A\cdot B)'\cdot B)' = A\cdot B + B' = A + B'$

$Z = (X\cdot Y)' = X' + Y' = A\cdot B' + A'\cdot B$

$F = (X\cdot Y\cdot Z)' = [(B + A')\cdot(A + B')\cdot(A\cdot B' + A'\cdot B)]' = [(A\cdot B + A'\cdot B')\cdot(A\cdot B' + A'\cdot B)]' = 0' = 1$

Q3a.

A	*B*	*F*
0	0	0
0	1	0
1	0	1
1	1	1

Q3b.

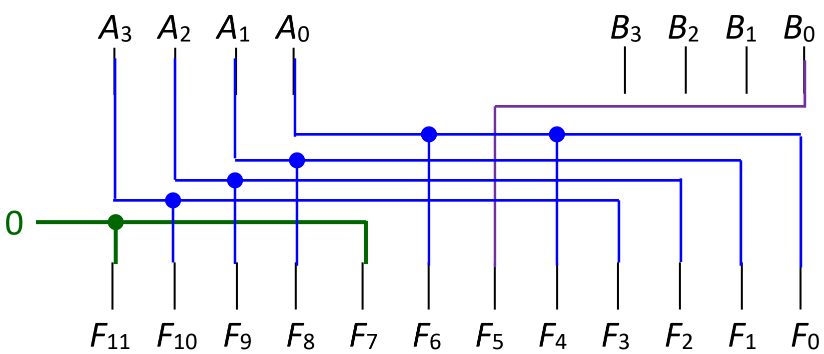
A	*B*	*C*	*D*	*G*
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

Q3c. $H = A\cdot B\cdot C'\cdot D' + A\cdot B\cdot C'\cdot D$
$= A\cdot B\cdot C'$

Q3d. Note that 50×A on A=0, 1, 2, 3, 4, …, 9 gives us 000, 051, 102, 153, 204, …, 459. It can be seen that the left-most digit 0, 0, 1, 1, 2, 2, 3, 3, 4, 4 is simply $A/2$. Hence the left-most 4 bits $F_{11}F_{10}F_9F_8$ are $0A_3A_2A_1$.

The middle digit is 0 (or 0000 in binary) if A is even (i.e. A_0= 0), or 5 (or 0101 in binary) if A is odd (i.e. A_0= 1). The right-most digit is A itself.

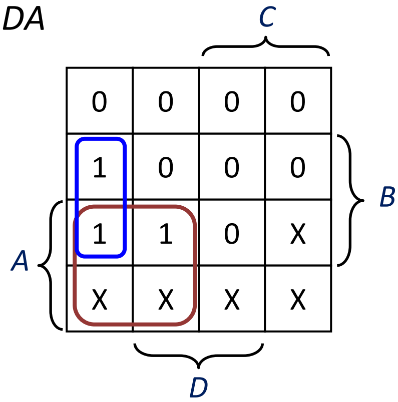
Finally, 20×(B%2) is achieved by putting B_0 into F_5.



Q2.

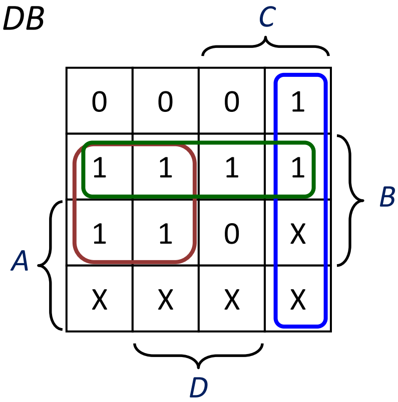
Current state				Nex state						
A	*B*	*C*	*D*	A^+	B^+	C^+	D^+	*TC*	*JD*	*KD*
0	0	0	0	0	0	0	1	0	1	X
0	0	0	1	0	0	1	1	1	X	0
0	0	1	0	0	1	1	0	0	0	X
0	0	1	1	0	0	1	0	0	X	1
0	1	0	0	1	1	0	0	0	0	X
0	1	0	1	0	1	0	0	0	X	1
0	1	1	0	0	1	1	1	0	1	X
0	1	1	1	0	1	0	1	1	X	0
1	0	0	0	X(1)	X(0,1)	X(0)	X(1)	X(0)	X(1)	X(0)
1	0	0	1	X(1)	X(0,1)	X(1)	X(1)	X(1)	X(1)	X(0)
1	0	1	0	X(0)	X(1)	X(1)	X(1)	X(0)	X(1)	X(1)
1	0	1	1	X(0)	X(0)	X(0)	X(0)	X(1)	X(1)	X(1)
1	1	0	0	1	1	0	1	0	1	X
1	1	0	1	1	1	1	1	1	X	0
1	1	1	0	X(0)	X(1)	X(1)	X(1)	X(0)	X(1)	X(1)
1	1	1	1	0	0	0	0	1	X	1

DA



DA = A·C' + B·C'·D'

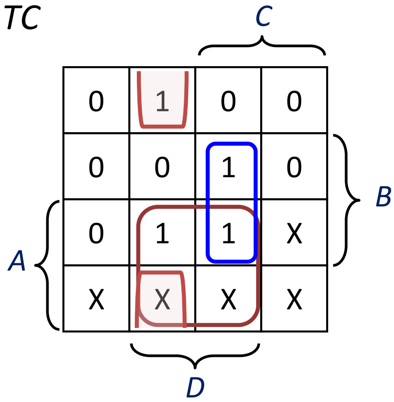
DB



DB = A'·B + C·D' + B·C'

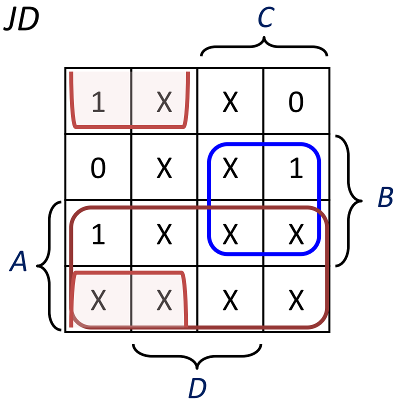
or *A'·B + C·D' + A·C'*

TC



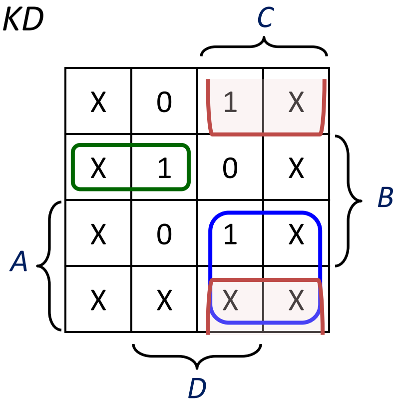
TC = A·D + B·C·D + B'·C'·D

JD



JD = A + B·C + B'·C'

KD



KD = A·C + B'·C + A'·B·C'

Q5. Tested on QTSpim

Data | Text

Text

```
                                            User Text Segment [00400000]..[00440000]
[00400000] 8fa40000  lw $4, 0($29)            ; 183: lw $a0 0($sp) # argc
[00400004] 27a50004  addiu $5, $29, 4         ; 184: addiu $a1 $sp 4 # argv
[00400008] 24a60004  addiu $6, $5, 4          ; 185: addiu $a2 $a1 4 # envp
[0040000c] 00041080  sll $2, $4, 2            ; 186: sll $v0 $a0 2
[00400010] 00c23021  addu $6, $6, $2          ; 187: addu $a2 $a2 $v0
[00400014] 0c100009  jal 0x00400024 [main]    ; 188: jal main
[00400018] 00000000  nop                      ; 189: nop
[0040001c] 3402000a  ori $2, $0, 10           ; 191: li $v0 10
[00400020] 0000000c  syscall                  ; 192: syscall # syscall 10 (exit)
[00400024] 3c101001  lui $16, 4097 [A]        ; 8: la $s0, A # $s0 is the base address of array A
[00400028] 3c011001  lui $1, 4097 [B]         ; 9: la $s1, B # $s1 is the base address of array B
[0040002c] 34310020  ori $17, $1, 32 [B]
[00400030] 3c011001  lui $1, 4097 [n]         ; 10: la $t0, n # $t0 is the address of n (size of array)
[00400034] 34280040  ori $8, $1, 64 [n]
[00400038] 8d120000  lw $18, 0($8)            ; 11: lw $s2, 0($t0) # $s2 is the content of n
[0040003c] 12400011  beq $18, $0, 68 [End-0x0040003c]
[00400040] 2258ffff  addi $24, $18, -1        ; 16: addi $t8, $s2, -1 # $t8 = n-1
[00400044] 0018c080  sll $24, $24, 2          ; 17: sll $t8, $t8, 2 # $t8 = 4*(n-1)
[00400048] 02184020  add $8, $16, $24         ; 18: add $t0, $s0, $t8
[0040004c] 02384820  add $9, $17, $24         ; 19: add $t1, $s1, $t8
[00400050] 8d0a0000  lw $10, 0($8)            ; 20: lw $t2, 0($t0) # $t2 = A[i]
[00400054] 8d2b0000  lw $11, 0($9)            ; 21: lw $t3, 0($t1) # $t3 = B[i]
[00400058] 316c0003  andi $12, $11, 3         ; 22: andi $t4, $t3, 3 # $t4 = B[i]%4
[0040005c] 218cfffd  addi $12, $12, -3        ; 23: addi $t4, $t4, -3 # $t4 = (B[i]%4)-3
[00400060] 11800003  beq $12, $0, 12 [A1-0x00400060]; 24: beq $t4, $zero, A1 # if (B[i]%4 == 3) goto A1
[00400064] 014b5020  add $10, $10, $11        ; 25: add $t2, $t2, $t3 # else A[i] += B[i]
[00400068] 0810001c  j 0x00400070 [A2]        ; 26: j A2
[0040006c] 214a0001  addi $10, $10, 1         ; 27: addi $t2, $t2, 1 # A[i]++
[00400070] ad0a0000  sw $10, 0($8)            ; 28: sw $t2, 0($t0)
[00400074] 2318fff8  addi $24, $24, -8        ; 29: addi $t8, $t8, -8 # i = i - 2
[00400078] 0300782a  slt $15, $24, $0         ; 30: slt $t7, $t8, $zero # if ($t8
[0040007c] 11e0fff3  beq $15, $0, -52 [Loop-0x0040007c]
[00400080] 3402000a  ori $2, $0, 10           ; 32: li $v0, 10 # system call code for exit
[00400084] 0000000c  syscall                  ; 33: syscall
```

Data: before and after

```
User data segment [10000000]..[10040000]
[10000000]..[1000ffff]  00000000
[10010000]    0000000011  0000000009  0000000031  0000000002
[10010010]    0000000009  0000000001  0000000006  0000000010
[10010020]    0000000003  0000000007  0000000002  0000000012
[10010030]    0000000011  0000000041  0000000019  0000000035
[10010040]    0000000008  0000000000  0000000000  0000000000
[10010050]..[1003ffff]  00000000
```

```
User data segment [10000000]..[10040000]
[10000000]..[1000ffff]  00000000
[10010000]    0000000011  0000000010  0000000031  0000000014
[10010010]    0000000009  0000000042  0000000006  0000000011
[10010020]    0000000003  0000000007  0000000002  0000000012
[10010030]    0000000011  0000000041  0000000019  0000000035
[10010040]    0000000008  0000000000  0000000000  0000000000
[10010050]..[1003ffff]  00000000
```

Q6. (c)

	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
I1 beq	F	D	E	M	W																									
I2 addi			F	D	E	M	W																							
I3 sll				F	D	E	M	W																						
I4 add					F	D	E	M	W																					
I5 add						F	D	E	M	W																				
I6 lw							F	D	E	M	W																			
I7 lw								F	D	E	M	W																		
I8 andi									F	D		E	M	W																
I9 addi										F	D		E	M	W															
I10 beq A1											F			D	E	M	W													
I11 add																														
I12 J A2																														
I13 A1: addi															F	D	E	M	W											
I14 A2: sw																F	D	E	M	W										
I15 addi																	F	D	E	M	W									
I16 slt																		F	D	E	M	W								
I17 beq																			F		D	E	M	W						

Q6. (d)

	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
I1 beq	F	D	E	M	W																									
I2 addi			F	D	E	M	W																							
I3 sll				F	D	E	M	W																						
I4 add					F	D	E	M	W																					
I5 add						F	D	E	M	W																				
I6 lw							F	D	E	M	W																			
I7 lw								F	D	E	M	W																		
I8 andi									F	D		E	M	W																
I9 addi										F	D		E	M	W															
I10 beq A1											F			D	E	M	W													
I11 add															F	D	E	M	W											
I12 J A2																F	D													
I13 A1: addi																														
I14 A2: sw																		F	D	W	M	W								
I15 addi																			F	D	E	M	W							
I16 slt																				F	D	E	M	W						
I17 beq																					F		D	E	M	W				

Q7. (a) Number of blocks = 128/4 = 32; Index: 5 bits; Offset: 4 bits

(b) *A*[0] at 0x10001000 → *A*[1023] at 0x10001FFC (because *A*[1024] at 0x10002000)

10001FFC → … 1111 1111 1100 → Index 31

B[0] at 0x1003F100 → *B*[1023] at 0x100400FC (because *B*[1024] at 0x10040100)

100400FC → … 0000 1111 1100 → Index 15

(c) 512 elements for each array are accessed: [1023], [1021], …, [3], [1].

A: 1024 accesses (1 read and 1 write per element); *B*: 512 accesses

(d) Since *A*[1023] and *B*[1023] are mapped to different indices, there will not be racing.

A: 256 misses, 768 hits, hit rate = 75%

B: 256 misses, 256 hits, hit rate = 50%

(e) 6147 instructions (3 + 512$\times$12). 9 misses, 6138 hits.

Index field: 3 bits; Offset: 3 bits

The first **beq** instruction is at addr 0x0040003c → … 0011 1100 → index 7, 2nd word

The following shows the cache content for the first iteration:

Index		
0	I2	I3
1	I4	I5
2	I6	I7
3	I8	I9
4	I10	
5		I13
6	I14	I15
7	I16	~~I1~~ I17

The misses are I1, I2, I4, I6, I8, I10, I13, I14 and I16. After that, all instructions in the loop are present in the cache.