CS2100 Computer Organization

AY20/21 Semester 1

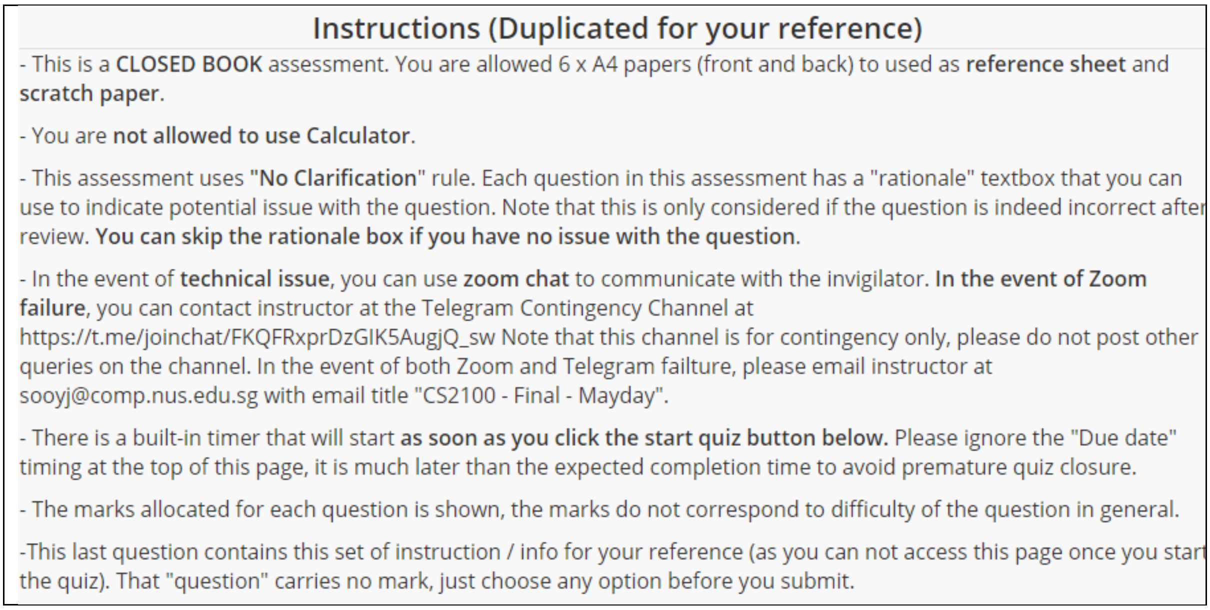
Final Assessment

Notes on format:

In November 2020, the CS2100 final assessment was conducted online on **Luminus Covid** as part of the Covid measure.

To enhance integrity of the assessment, questions are randomized in order. Some questions have multiple variants with equivalent difficulty.

This sample paper only capture a certain order and a subset of the variants.



Instructions (Duplicated for your reference)

- This is a **CLOSED BOOK** assessment. You are allowed 6 x A4 papers (front and back) to used as **reference sheet** and **scratch paper**.

- You are **not allowed to use Calculator**.

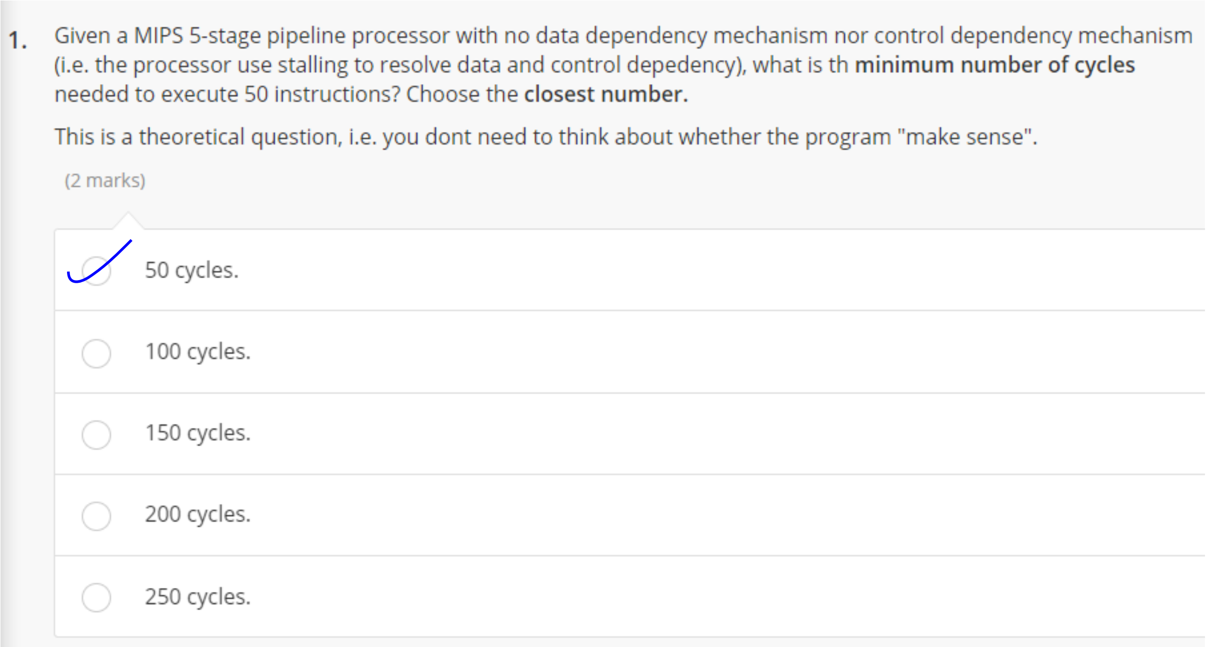
- This assessment uses **"No Clarification"** rule. Each question in this assessment has a "rationale" textbox that you can use to indicate potential issue with the question. Note that this is only considered if the question is indeed incorrect afte review. **You can skip the rationale box if you have no issue with the question.**

- In the event of **technical issue**, you can use **zoom chat** to communicate with the invigilator. **In the event of Zoom failure**, you can contact instructor at the Telegram Contingency Channel at https://t.me/joinchat/FKQFRxprDzGlK5AugjQ_sw Note that this channel is for contingency only, please do not post other queries on the channel. In the event of both Zoom and Telegram failture, please email instructor at sooyj@comp.nus.edu.sg with email title "CS2100 - Final - Mayday".

- There is a built-in timer that will start **as soon as you click the start quiz button below.** Please ignore the "Due date" timing at the top of this page, it is much later than the expected completion time to avoid premature quiz closure.

- The marks allocated for each question is shown, the marks do not correspond to difficulty of the question in general.

-This last question contains this set of instruction / info for your reference (as you can not access this page once you star the quiz). That "question" carries no mark, just choose any option before you submit.

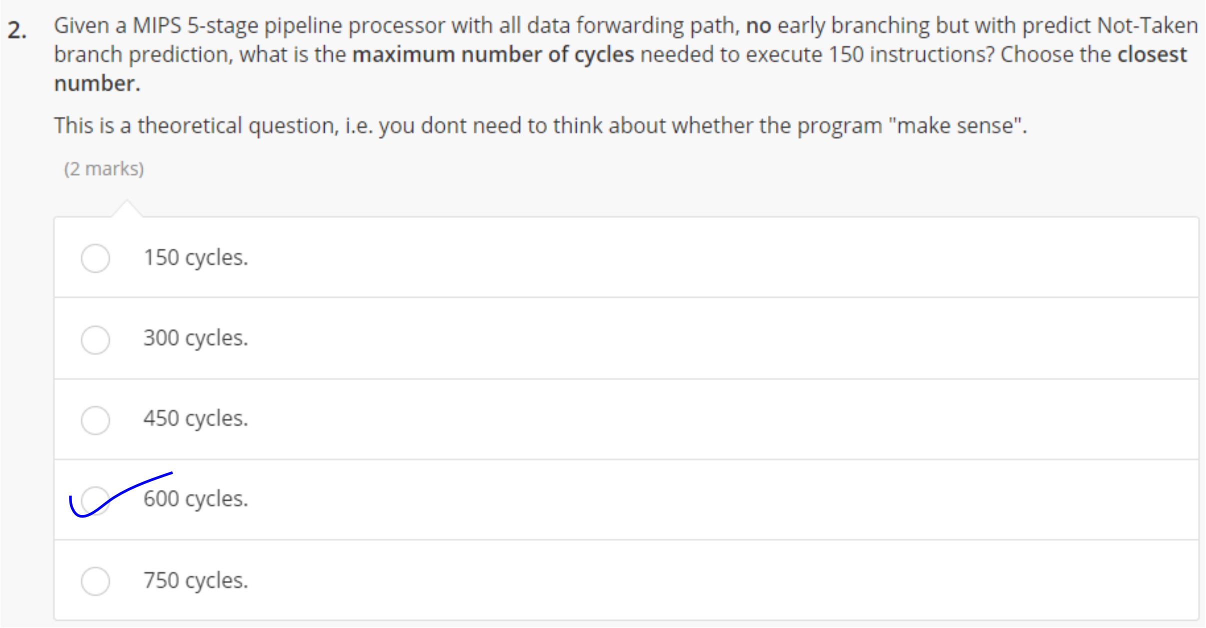


1. Given a MIPS 5-stage pipeline processor with no data dependency mechanism nor control dependency mechanism (i.e. the processor use stalling to resolve data and control depedency), what is th **minimum number of cycles** needed to execute 50 instructions? Choose the **closest number.**

This is a theoretical question, i.e. you dont need to think about whether the program "make sense".

(2 marks)

- [✓ ○ 50 cycles.]
- ○ 100 cycles.
- ○ 150 cycles.
- ○ 200 cycles.
- ○ 250 cycles.

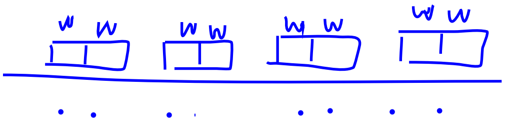


2. Given a MIPS 5-stage pipeline processor with all data forwarding path, **no** early branching but with predict Not-Taken branch prediction, what is the **maximum number of cycles** needed to execute 150 instructions? Choose the **closest number.**

This is a theoretical question, i.e. you dont need to think about whether the program "make sense".

(2 marks)

- ○ 150 cycles.
- ○ 300 cycles.
- ○ 450 cycles.
- [✓ ○ 600 cycles.]
- ○ 750 cycles.



2^7

3. Suppose we have a 4-Way Set Associative cache with 16KB (KB = 2^{10} Bytes) data capacity and 512 Sets. During a memory load, a cache miss occurs at Cache Set Index 29 while looking for memory tag **0x2233**.

0... 02233|29|000|

What is the **32-bit** memory address of the **memory block to be loaded into the cache** from memory?

Give your answer in hexadecimal with **capitalized A, B, C... F. Leave out the 0x prefix.**

(4 marks)

4. Given F(A, B, C, D, E), **a sum term of two literals**, e.g. (B + E') can be expanded into how many **maxterms**?

(2 marks)

[✓ ○ 3]

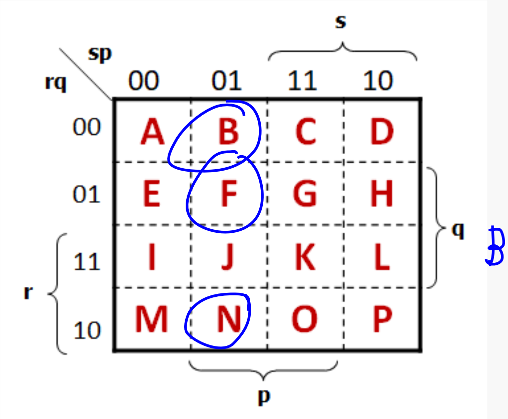
○ 6

○ 8

○ 16

○ All other options are incorrect.

5. Mr.Snoozy slept through the K-Map lecture and "anyhow" drew the following K-Map for function F(p, q, r, s):



rq \ sp	00	01	11	10
00	A	B	C	D
01	E	F	G	H
11	I	J	K	L
10	M	N	O	P

BF N

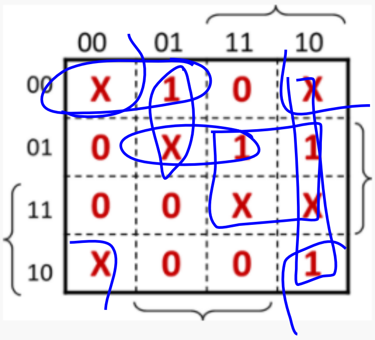
He asked you to help filling in the minterms for the expression **p.q'.r.s' + p.r'.s'**. Give the alpabet(s) of all K-Map cells that contains a '1'.

Please give you answer in alphabetical order with no space, e.g. "**CGKO**" for all 4 cells in the 3rd column.

(3 marks)

K-Map II

Observe the following K-Map carefully:



	00	01	11	10
00	X	1	0	X
01	0	X	1	1
11	0	0	X	X
10	X	0	0	1

X = Dont Care

6. Fill in the blanks

(3 marks)

Give the following for the K-Map:

There are __1__ Prime Implicants (PI).

There are __2__ Essential Prime Implicants (EPI).

Enter the correct answer below.

1 6

2 0/1

Performance

7. Suppose a program **P** consists of:

- Type **A** instruction with CPI of 3 cycles
- Type **B** instruction with CPI of 2 cycles
- There are exactly 50% of each type of instructions in **P**

Select all improvements below that result in **exactly 25% improvement in execution time**. e.g. if the original program P takes 20 seconds, the improvement should result in the new execution time of 16 seconds (20 / 16 = 1.25).

Each option is **independent from each other, i.e. should be considered on its own only.**

(2 marks)

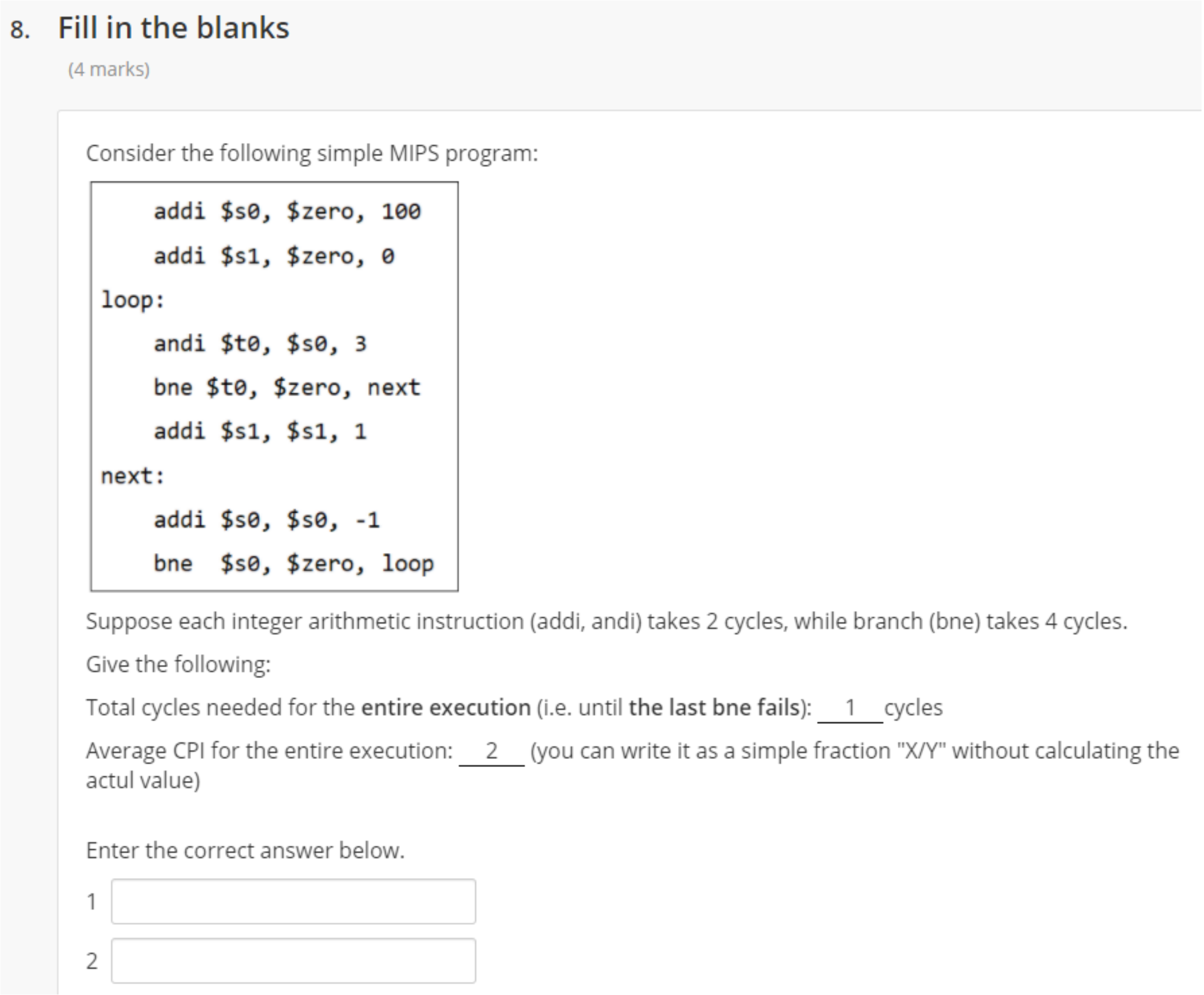
☐ Reduce the CPI of Type A instruction from 3 cycles to 2 cycles.

☐ All other options are incorrect. (You should not choose any other option if you select this).

☐ Reduce the CPI of Type B instruction from 2 cycles to 1 cycles.

☑ Change the program composition to 25% type A and 75% type B.

☐ Change the program composition to 75% type A and 25% type B.



8. Fill in the blanks

(4 marks)

Consider the following simple MIPS program:

```
    addi $s0, $zero, 100
    addi $s1, $zero, 0
loop:
    andi $t0, $s0, 3
    bne $t0, $zero, next
    addi $s1, $s1, 1
next:
    addi $s0, $s0, -1
    bne  $s0, $zero, loop
```

Suppose each integer arithmetic instruction (addi, andi) takes 2 cycles, while branch (bne) takes 4 cycles.

Give the following:

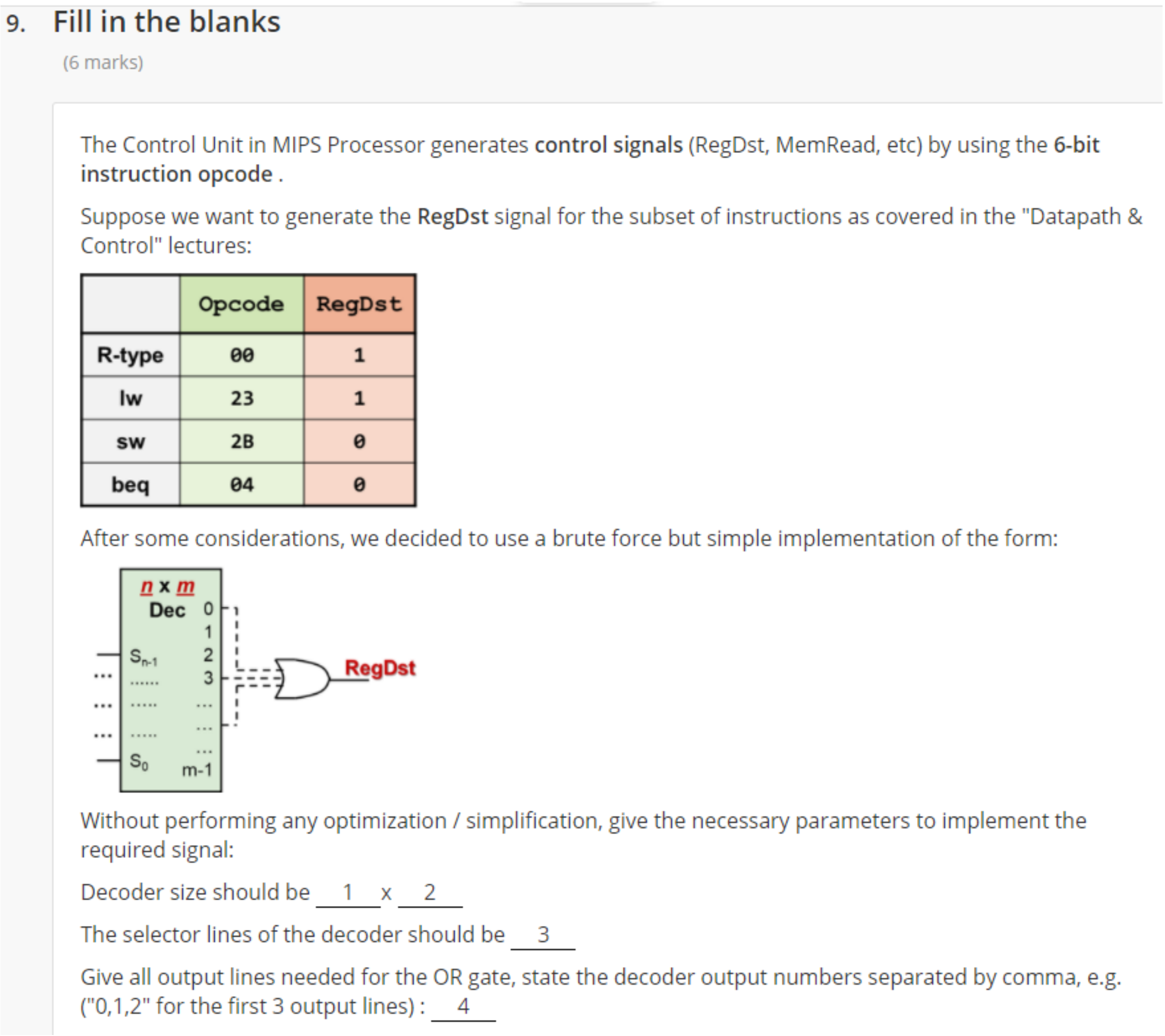
Total cycles needed for the **entire execution** (i.e. until **the last bne fails**): __1__ cycles

Average CPI for the entire execution: __2__ (you can write it as a simple fraction "X/Y" without calculating the actul value)

Enter the correct answer below.

1 [ ]

2 [ ]



9. Fill in the blanks

(6 marks)

The Control Unit in MIPS Processor generates **control signals** (RegDst, MemRead, etc) by using the **6-bit instruction opcode** .

Suppose we want to generate the **RegDst** signal for the subset of instructions as covered in the "Datapath & Control" lectures:

	Opcode	RegDst
R-type	00	1
lw	23	1
sw	2B	0
beq	04	0

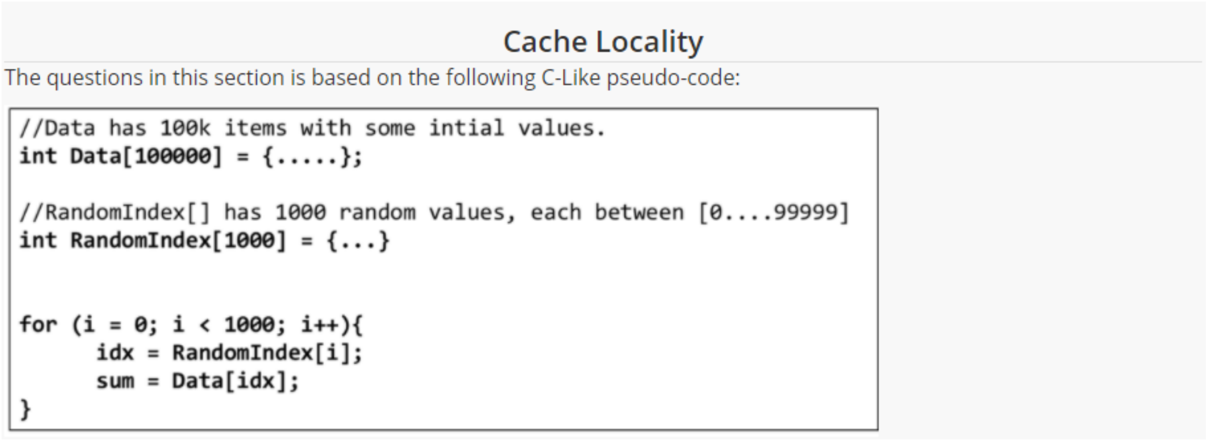
After some considerations, we decided to use a brute force but simple implementation of the form:

Without performing any optimization / simplification, give the necessary parameters to implement the required signal:

Decoder size should be 1 x 2

The selector lines of the decoder should be 3

Give all output lines needed for the OR gate, state the decoder output numbers separated by comma, e.g. ("0,1,2" for the first 3 output lines) : 4



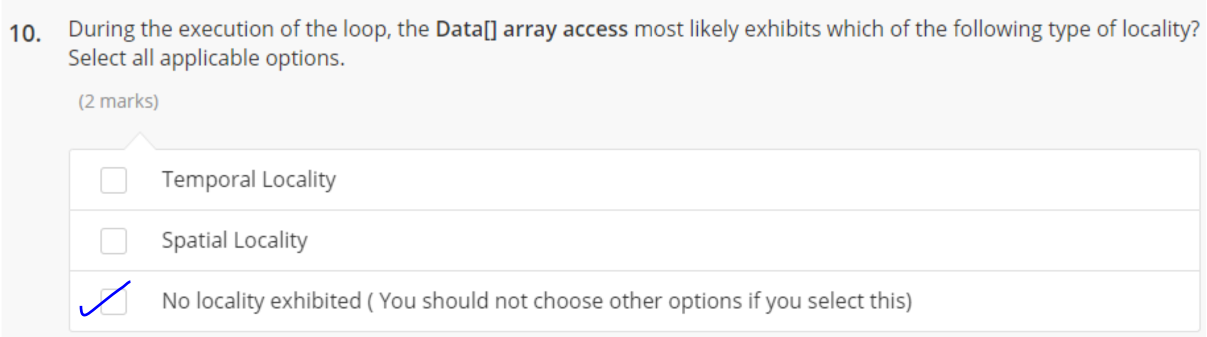
Cache Locality

The questions in this section is based on the following C-Like pseudo-code:

```
//Data has 100k items with some intial values.
int Data[100000] = {.....};

//RandomIndex[] has 1000 random values, each between [0....99999]
int RandomIndex[1000] = {...}


for (i = 0; i < 1000; i++){
      idx = RandomIndex[i];
      sum = Data[idx];
}
```



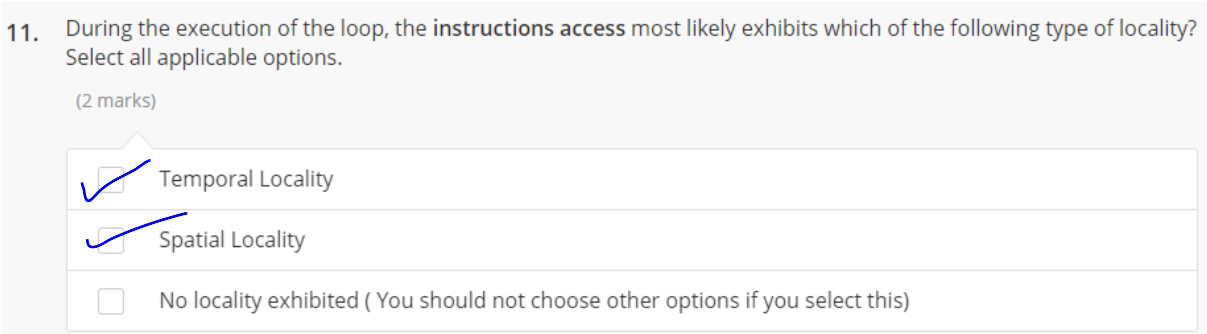
10. During the execution of the loop, the **Data[] array access** most likely exhibits which of the following type of locality? Select all applicable options.

(2 marks)

□ Temporal Locality

□ Spatial Locality

☑ No locality exhibited (You should not choose other options if you select this)



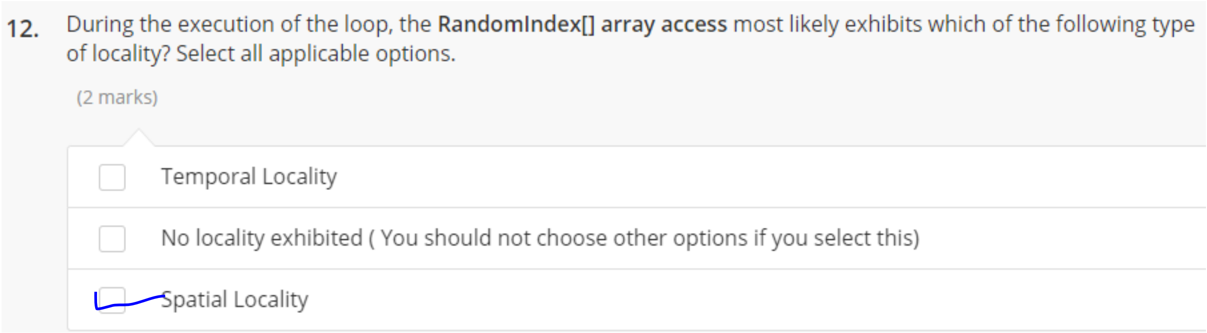
11. During the execution of the loop, the **instructions access** most likely exhibits which of the following type of locality? Select all applicable options.

(2 marks)

☑ Temporal Locality

☑ Spatial Locality

□ No locality exhibited (You should not choose other options if you select this)



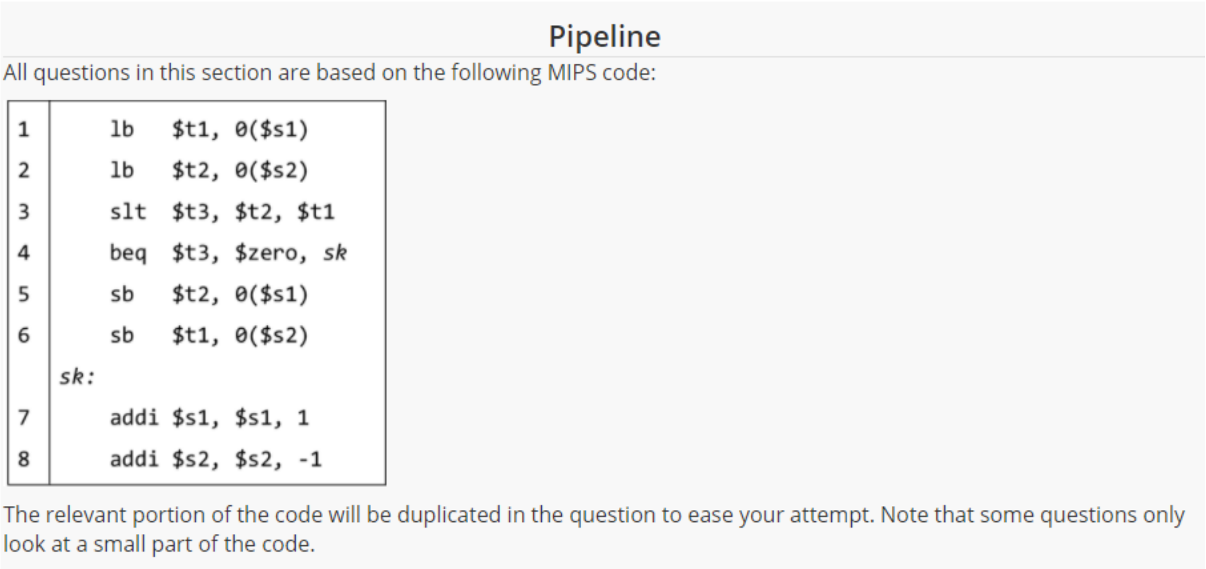
12. During the execution of the loop, the **RandomIndex[] array access** most likely exhibits which of the following type of locality? Select all applicable options.

(2 marks)

□ Temporal Locality

□ No locality exhibited (You should not choose other options if you select this)

☑ Spatial Locality

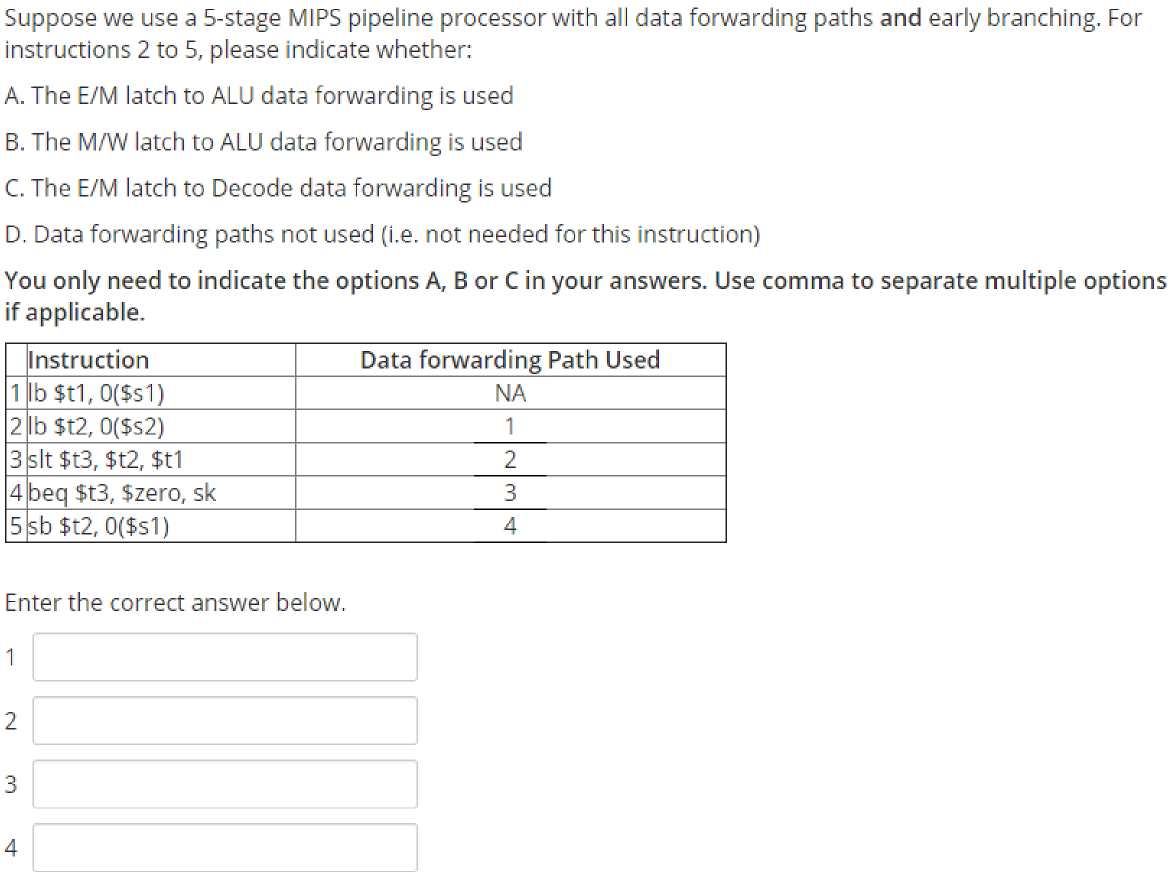


Pipeline

All questions in this section are based on the following MIPS code:

```
1       lb   $t1, 0($s1)
2       lb   $t2, 0($s2)
3       slt  $t3, $t2, $t1
4       beq  $t3, $zero, sk
5       sb   $t2, 0($s1)
6       sb   $t1, 0($s2)
    sk:
7       addi $s1, $s1, 1
8       addi $s2, $s2, -1
```

The relevant portion of the code will be duplicated in the question to ease your attempt. Note that some questions only look at a small part of the code.



Suppose we use a 5-stage MIPS pipeline processor with all data forwarding paths **and** early branching. For instructions 2 to 5, please indicate whether:

A. The E/M latch to ALU data forwarding is used

B. The M/W latch to ALU data forwarding is used

C. The E/M latch to Decode data forwarding is used

D. Data forwarding paths not used (i.e. not needed for this instruction)

You only need to indicate the options A, B or C in your answers. Use comma to separate multiple options if applicable.

	Instruction	Data forwarding Path Used
1	lb $t1, 0($s1)	NA
2	lb $t2, 0($s2)	1
3	slt $t3, $t2, $t1	2
4	beq $t3, $zero, sk	3
5	sb $t2, 0($s1)	4

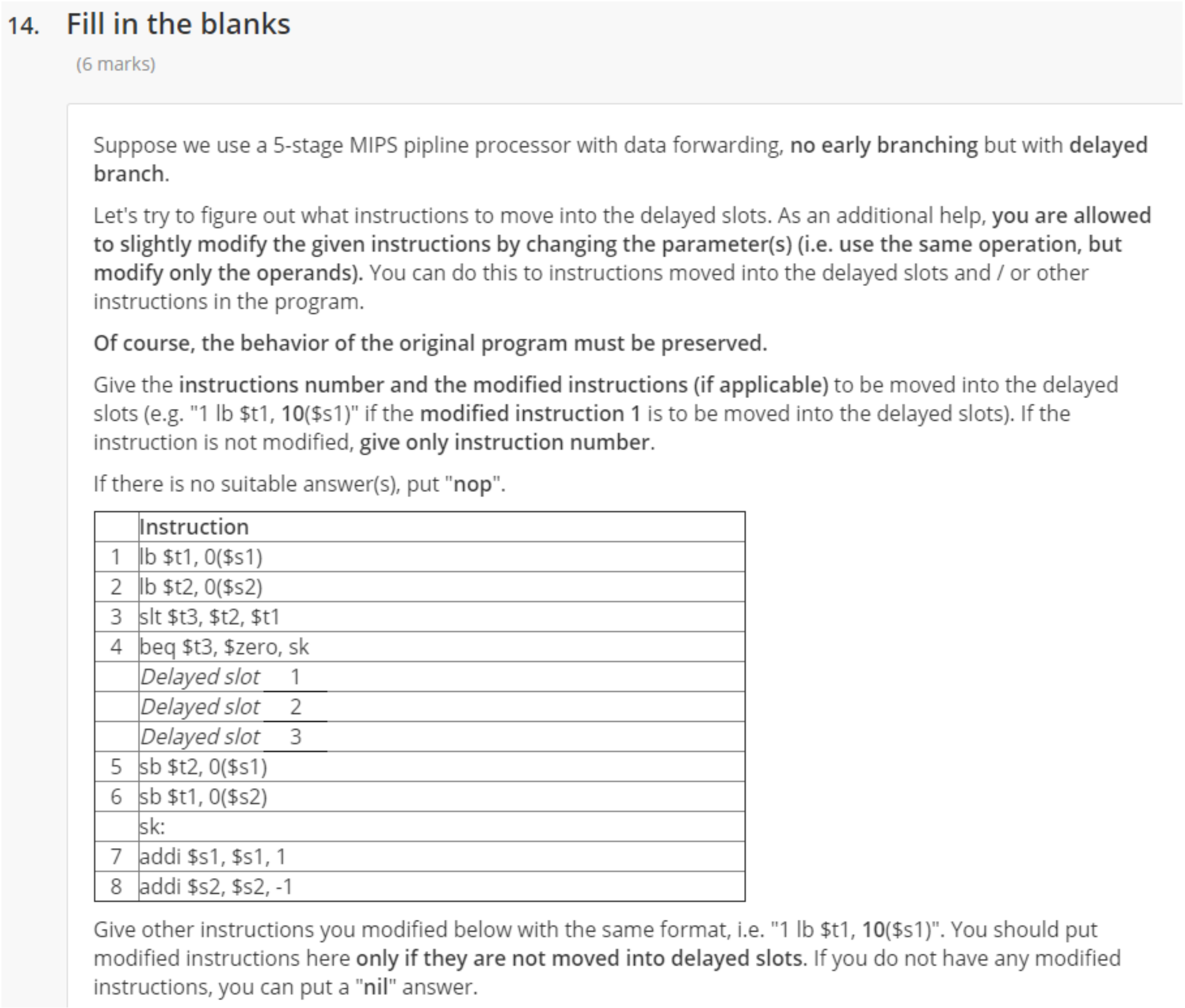
Enter the correct answer below.

1

2

3

4



14. Fill in the blanks

(6 marks)

Suppose we use a 5-stage MIPS pipline processor with data forwarding, **no early branching** but with **delayed branch.**

Let's try to figure out what instructions to move into the delayed slots. As an additional help, **you are allowed to slightly modify the given instructions by changing the parameter(s) (i.e. use the same operation, but modify only the operands).** You can do this to instructions moved into the delayed slots and / or other instructions in the program.

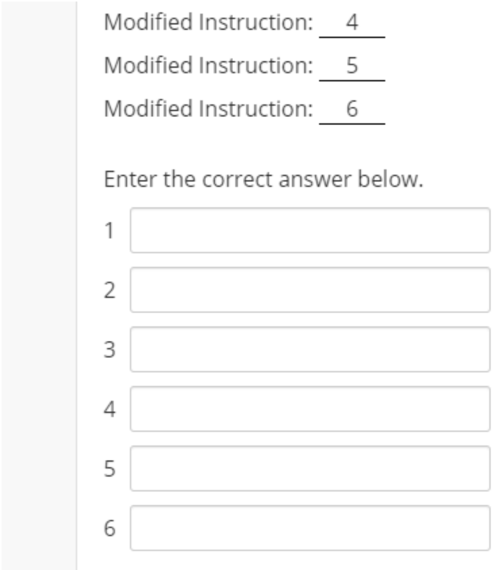
Of course, the behavior of the original program must be preserved.

Give the **instructions number and the modified instructions (if applicable)** to be moved into the delayed slots (e.g. "1 lb $t1, **10**($s1)" if the **modified instruction 1** is to be moved into the delayed slots). If the instruction is not modified, **give only instruction number**.

If there is no suitable answer(s), put "**nop**".

	Instruction
1	lb $t1, 0($s1)
2	lb $t2, 0($s2)
3	slt $t3, $t2, $t1
4	beq $t3, $zero, sk
	Delayed slot 1
	Delayed slot 2
	Delayed slot 3
5	sb $t2, 0($s1)
6	sb $t1, 0($s2)
	sk:
7	addi $s1, $s1, 1
8	addi $s2, $s2, -1

Give other instructions you modified below with the same format, i.e. "1 lb $t1, **10**($s1)". You should put modified instructions here **only if they are not moved into delayed slots**. If you do not have any modified instructions, you can put a "**nil**" answer.



Modified Instruction: 4

Modified Instruction: 5

Modified Instruction: 6

Enter the correct answer below.

1

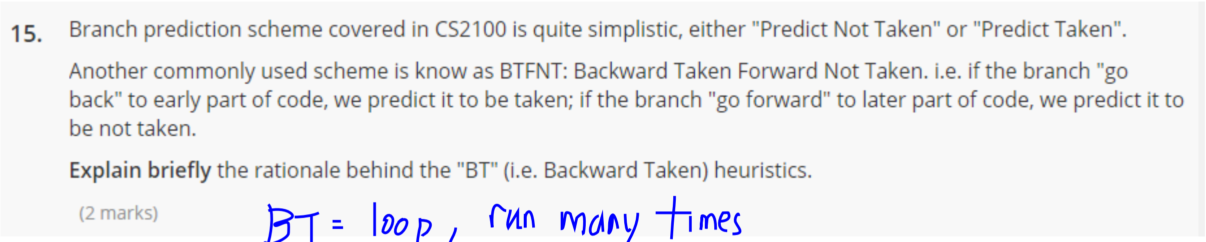
2

3

4

5

6



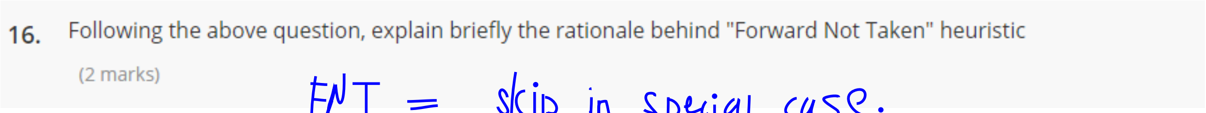
15. Branch prediction scheme covered in CS2100 is quite simplistic, either "Predict Not Taken" or "Predict Taken".

Another commonly used scheme is know as BTFNT: Backward Taken Forward Not Taken. i.e. if the branch "go back" to early part of code, we predict it to be taken; if the branch "go forward" to later part of code, we predict it to be not taken.

Explain briefly the rationale behind the "BT" (i.e. Backward Taken) heuristics.

(2 marks)

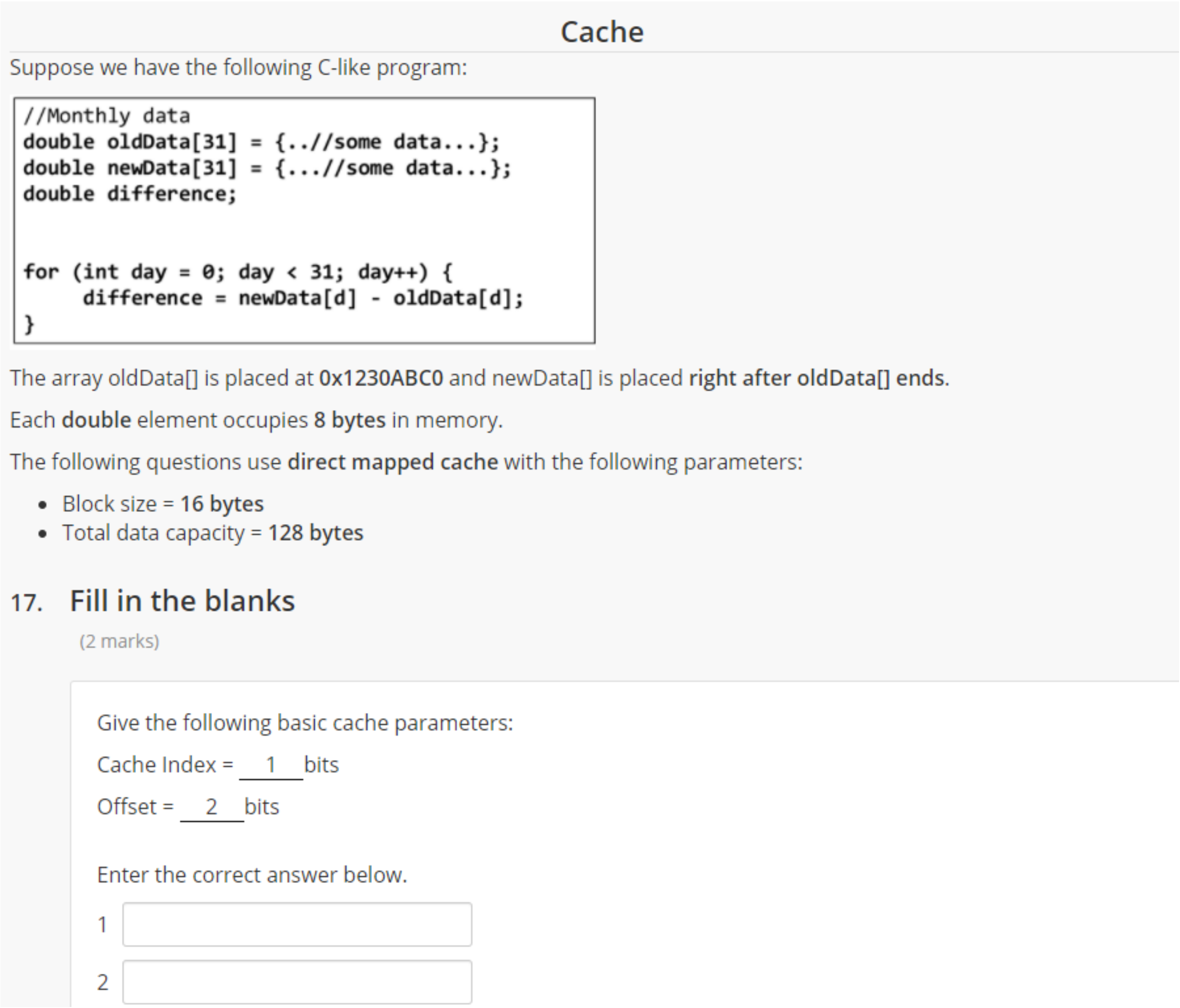
BT = loop , run many times



16. Following the above question, explain briefly the rationale behind "Forward Not Taken" heuristic

(2 marks)

FNT = skip in special case.



Cache

Suppose we have the following C-like program:

```
//Monthly data
double oldData[31] = {..//some data...};
double newData[31] = {...//some data...};
double difference;


for (int day = 0; day < 31; day++) {
     difference = newData[d] - oldData[d];
}
```

The array oldData[] is placed at **0x1230ABC0** and newData[] is placed **right after oldData[] ends**.

Each **double** element occupies **8 bytes** in memory.

The following questions use **direct mapped cache** with the following parameters:

- Block size = **16 bytes**
- Total data capacity = **128 bytes**

17. Fill in the blanks

(2 marks)

Give the following basic cache parameters:

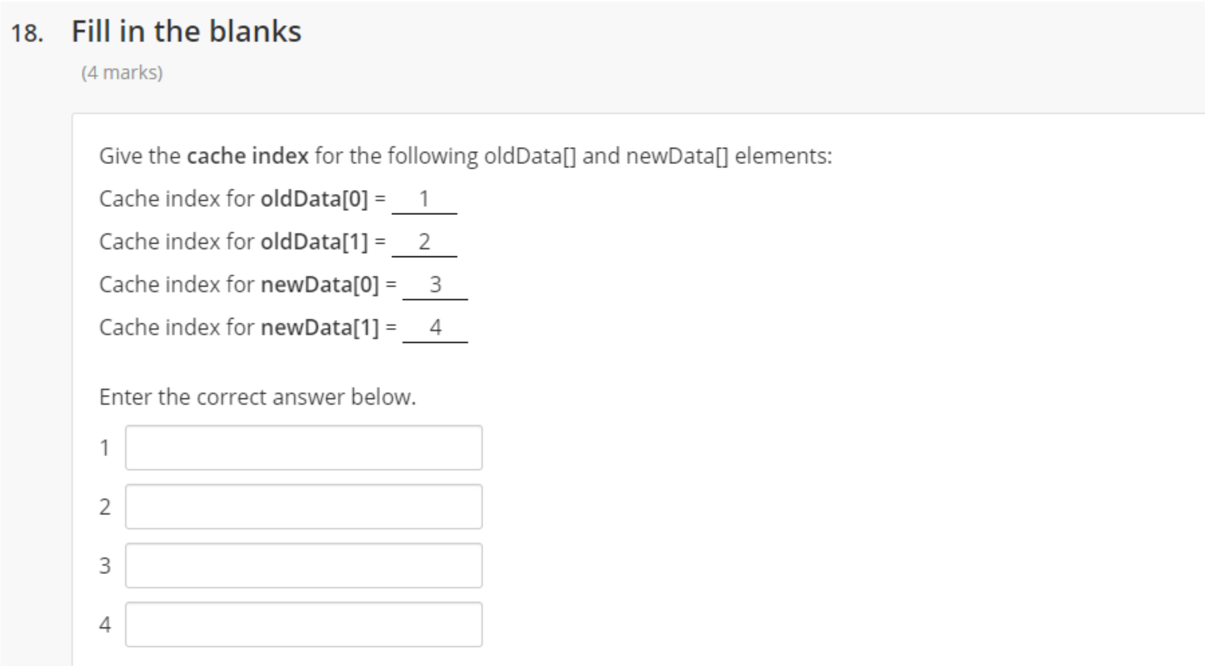
Cache Index = __1__ bits

Offset = __2__ bits

Enter the correct answer below.

1

2



18. Fill in the blanks

(4 marks)

Give the **cache index** for the following oldData[] and newData[] elements:

Cache index for **oldData[0]** = __1__

Cache index for **oldData[1]** = __2__

Cache index for **newData[0]** = __3__

Cache index for **newData[1]** = __4__

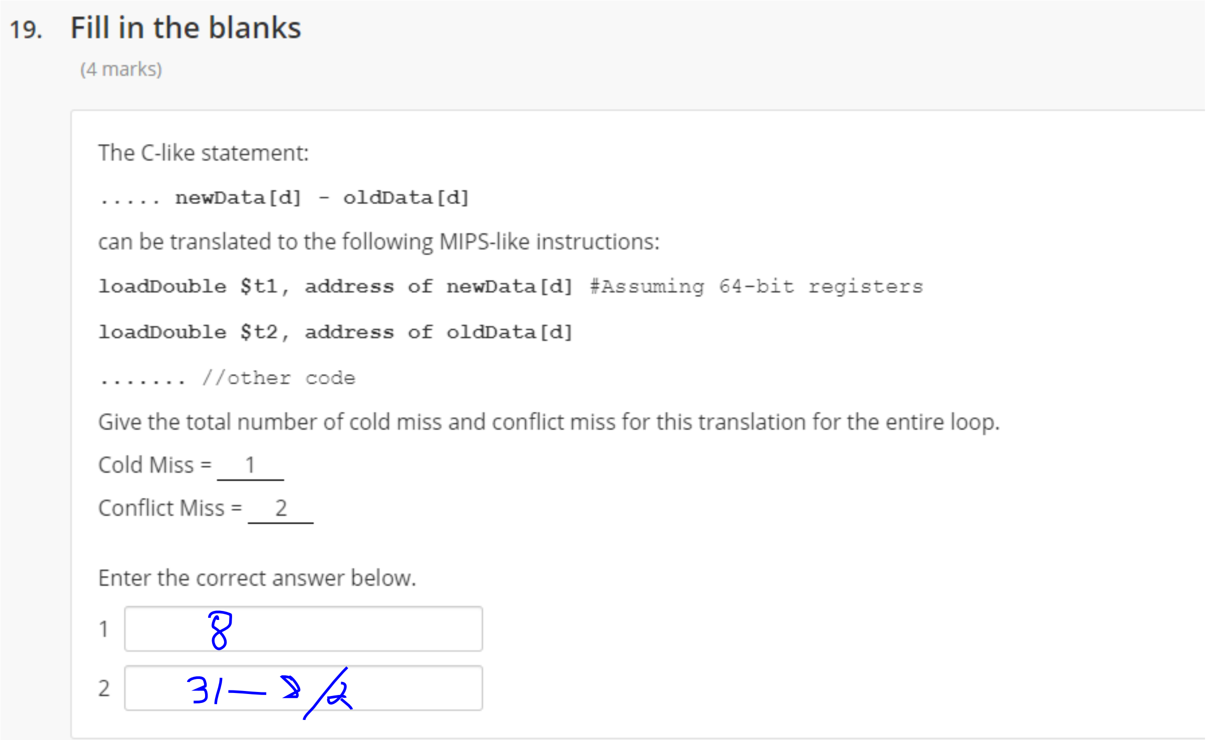
Enter the correct answer below.

1

2

3

4



19. Fill in the blanks

(4 marks)

The C-like statement:

```
..... newData[d] - oldData[d]
```

can be translated to the following MIPS-like instructions:

```
loadDouble $t1, address of newData[d] #Assuming 64-bit registers
loadDouble $t2, address of oldData[d]
....... //other code
```

Give the total number of cold miss and conflict miss for this translation for the entire loop.

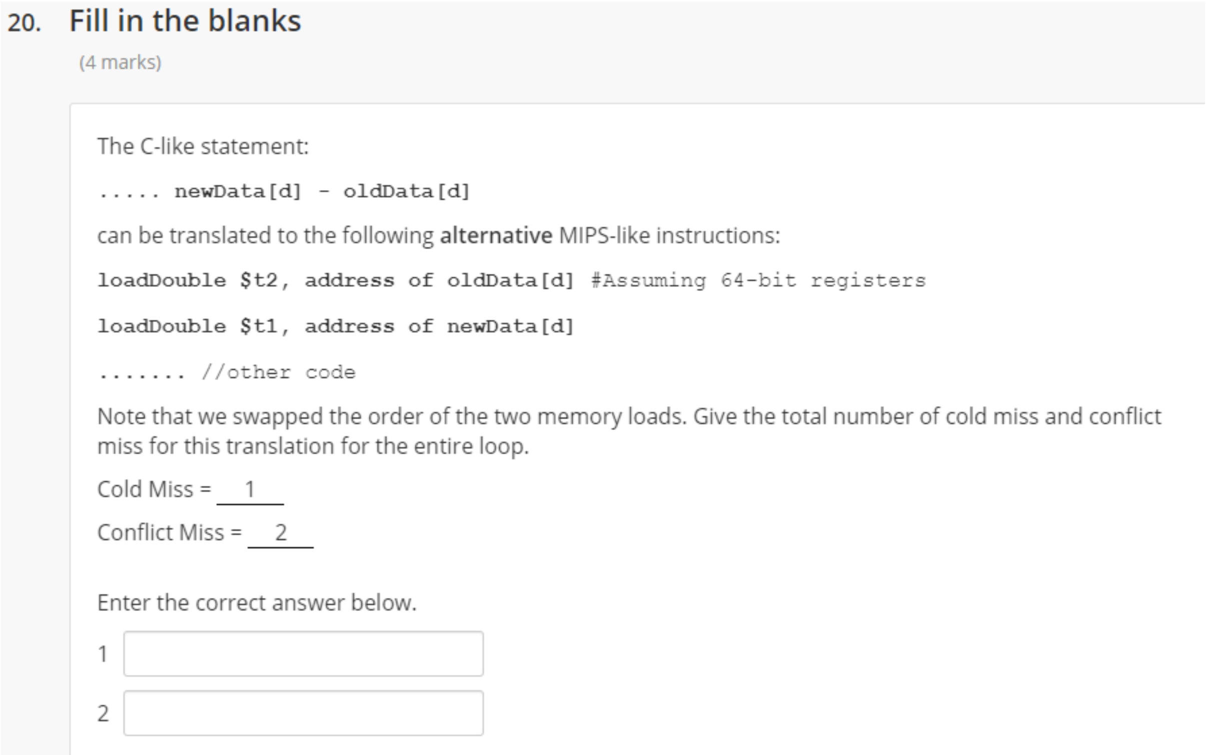
Cold Miss = __1__

Conflict Miss = __2__

Enter the correct answer below.

1 8

2 31—2/2



20. Fill in the blanks

(4 marks)

The C-like statement:

```
..... newData[d] - oldData[d]
```

can be translated to the following **alternative** MIPS-like instructions:

```
loadDouble $t2, address of oldData[d] #Assuming 64-bit registers

loadDouble $t1, address of newData[d]

....... //other code
```

Note that we swapped the order of the two memory loads. Give the total number of cold miss and conflict miss for this translation for the entire loop.

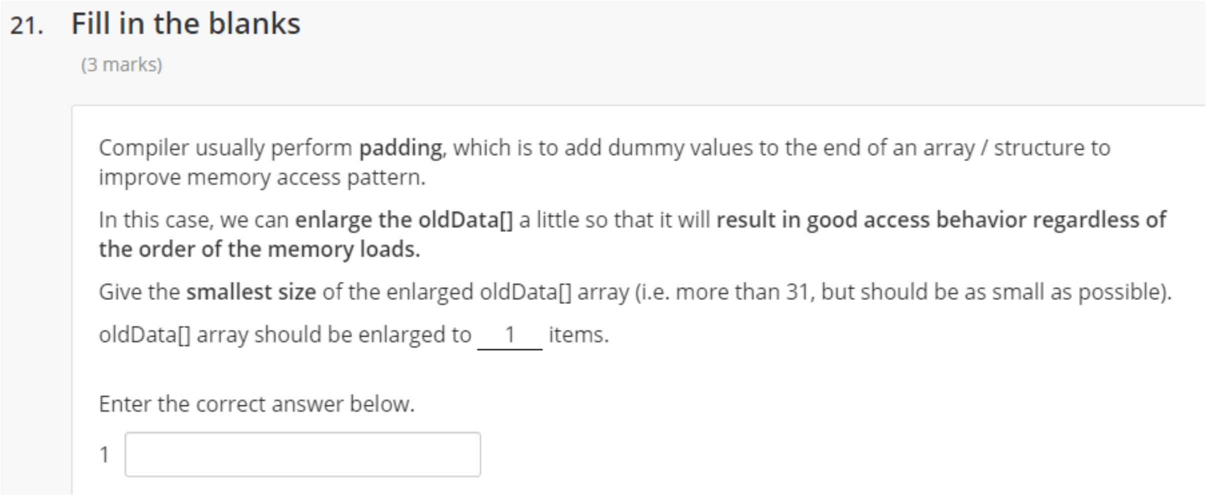
Cold Miss = __1__

Conflict Miss = __2__

Enter the correct answer below.

1 □

2 □



21. Fill in the blanks

(3 marks)

Compiler usually perform **padding**, which is to add dummy values to the end of an array / structure to improve memory access pattern.

In this case, we can **enlarge the oldData[]** a little so that it will **result in good access behavior regardless of the order of the memory loads.**

Give the **smallest size** of the enlarged oldData[] array (i.e. more than 31, but should be as small as possible).

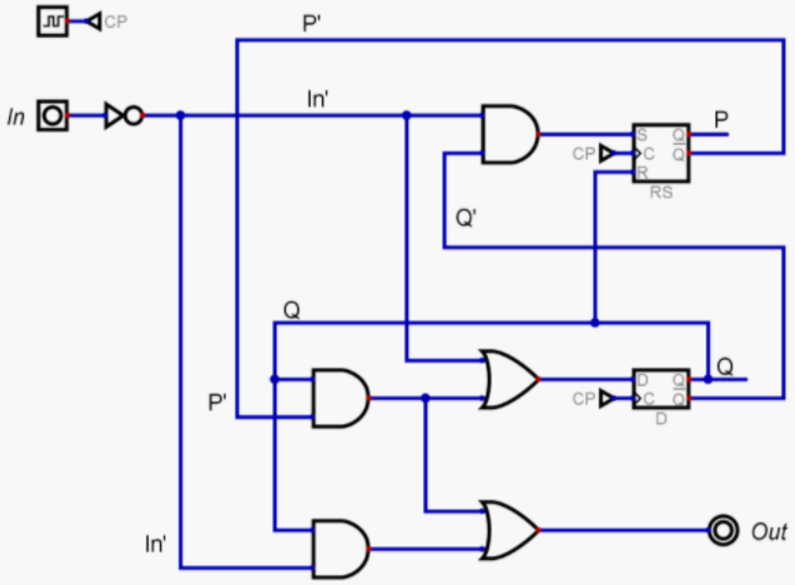
oldData[] array should be enlarged to __1__ items.

Enter the correct answer below.

1 □

Sequential Analysis

Given the following Sequential Circuit:



The state variable are **P** (produced by SR flipflop) and **Q** (produced by D flipflop). There is an input variable **In** and a output variable **Out** from the circuit. Analyse the behavior of the circuit.

Take time to do all the necessary steps carefully. There are questions along the way to check various aspects of your work to give partial credit even if you cant solve the questions entirely.

22. Give the flipflop input function for **SP**.

(2 marks)

23. Give the **simplified** POS for the "**Out**" variable.

(3 marks)

24. Fill in the blanks

(4 marks)

Give the next state and output value for **state PQ = 00.**

When In = 0, Next state P^+Q^+ = __1__; Output = __2__

When In = 1, Next state P^+Q^+ = __3__; Output = __4__

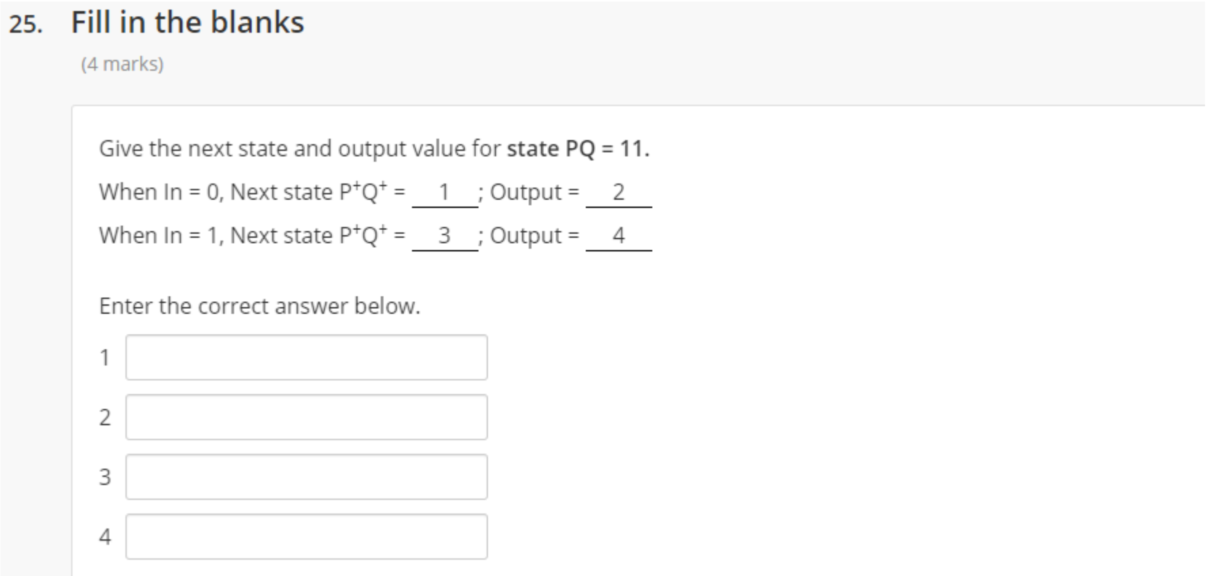
Enter the correct answer below.

1 ▭

2 ▭

3 ▭

4 ▭



25. **Fill in the blanks**

(4 marks)

Give the next state and output value for **state PQ = 11.**

When In = 0, Next state P^+Q^+ = __1__; Output = __2__

When In = 1, Next state P^+Q^+ = __3__; Output = __4__

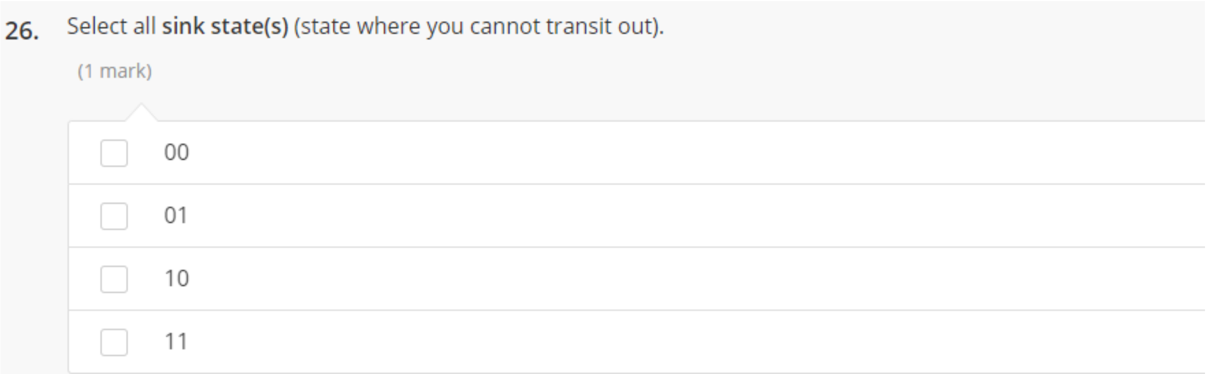
Enter the correct answer below.

1

2

3

4



26. Select all **sink state(s)** (state where you cannot transit out).

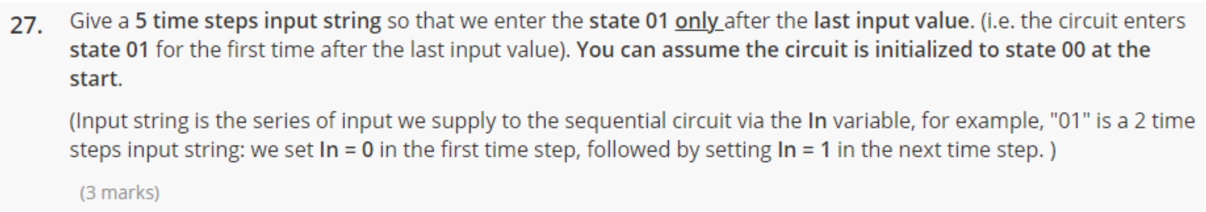
(1 mark)

□ 00

□ 01

□ 10

□ 11



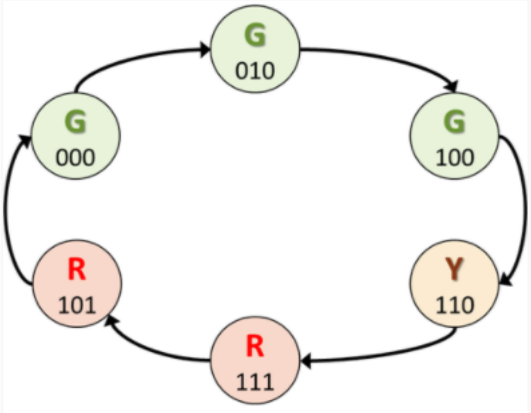
27. Give a **5 time steps input string** so that we enter the **state 01** <u>only</u> after the **last input value**. (i.e. the circuit enters **state 01** for the first time after the last input value). **You can assume the circuit is initialized to state 00 at the start.**

(Input string is the series of input we supply to the sequential circuit via the **In** variable, for example, "01" is a 2 time steps input string: we set **In = 0** in the first time step, followed by setting **In = 1** in the next time step.)

(3 marks)

Sequential Circuit Construction

Traffic light is probably the most well known sequential circuit, let's try to construct one, shall we?



Above is the specification, each state has a "Color" (G=Green, Y=Yellow, R=Red). Pay attention to the state values **ABC** below each of the color label. **We assume the circuit will start in state ABC = 000.**

You are asked to construct a sequential circuit with the above state transition behavior using:

- JK flip flop for state variable A
- D flip flop for state variable B
- T flip flop for state variable C

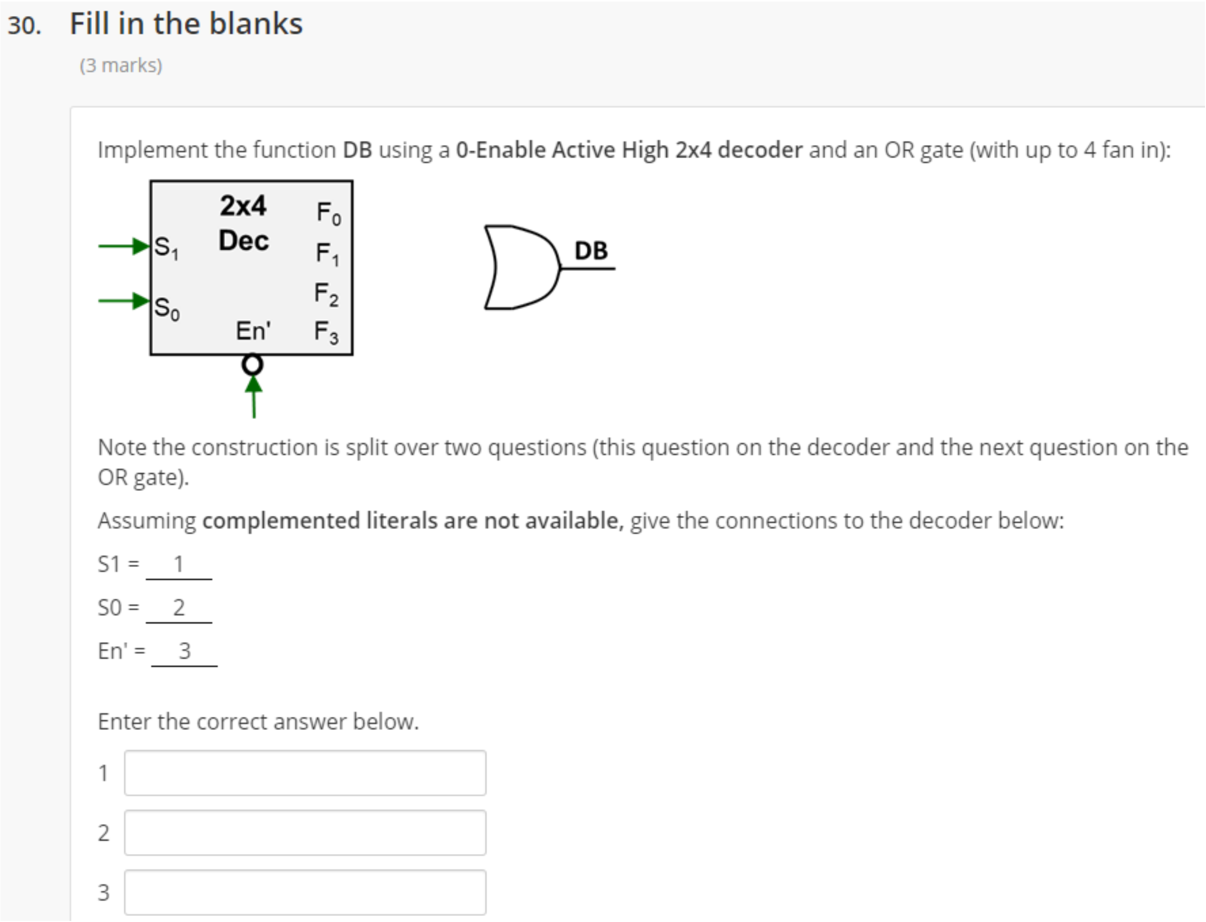
Please take time to carry out the steps and keep your working on scratch paper. We have questions to check the steps along the way.

28. Give the simplified SOP for **KA.**

(3 marks)

29. Give the simplifed SOP for **DB.**

(3 marks)



30. **Fill in the blanks**

(3 marks)

Implement the function **DB** using a **0-Enable Active High 2x4 decoder** and an OR gate (with up to 4 fan in):

Note the construction is split over two questions (this question on the decoder and the next question on the OR gate).

Assuming **complemented literals are not available,** give the connections to the decoder below:

S1 = __1__

S0 = __2__

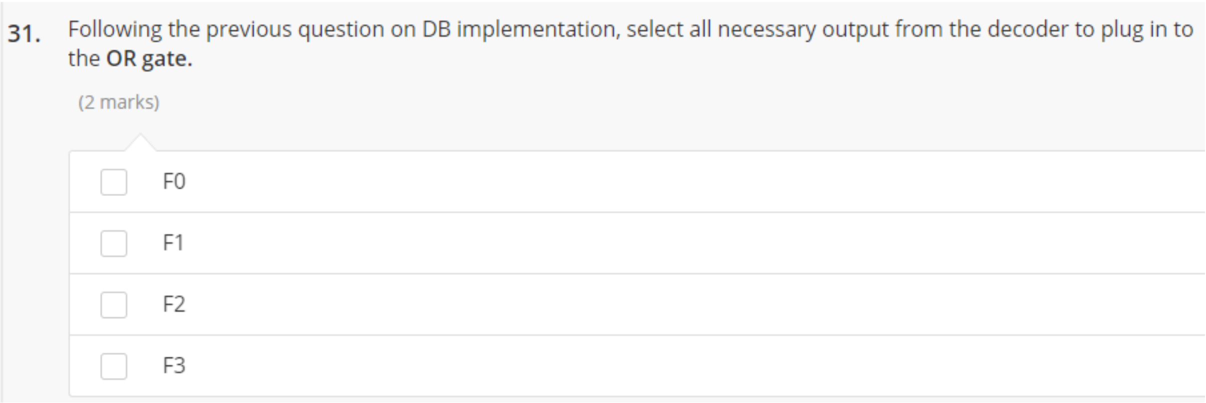
En' = __3__

Enter the correct answer below.

1 ▭

2 ▭

3 ▭



31. Following the previous question on DB implementation, select all necessary output from the decoder to plug in to the **OR gate.**

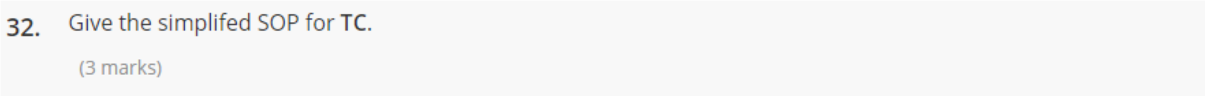
(2 marks)

☐ F0

☐ F1

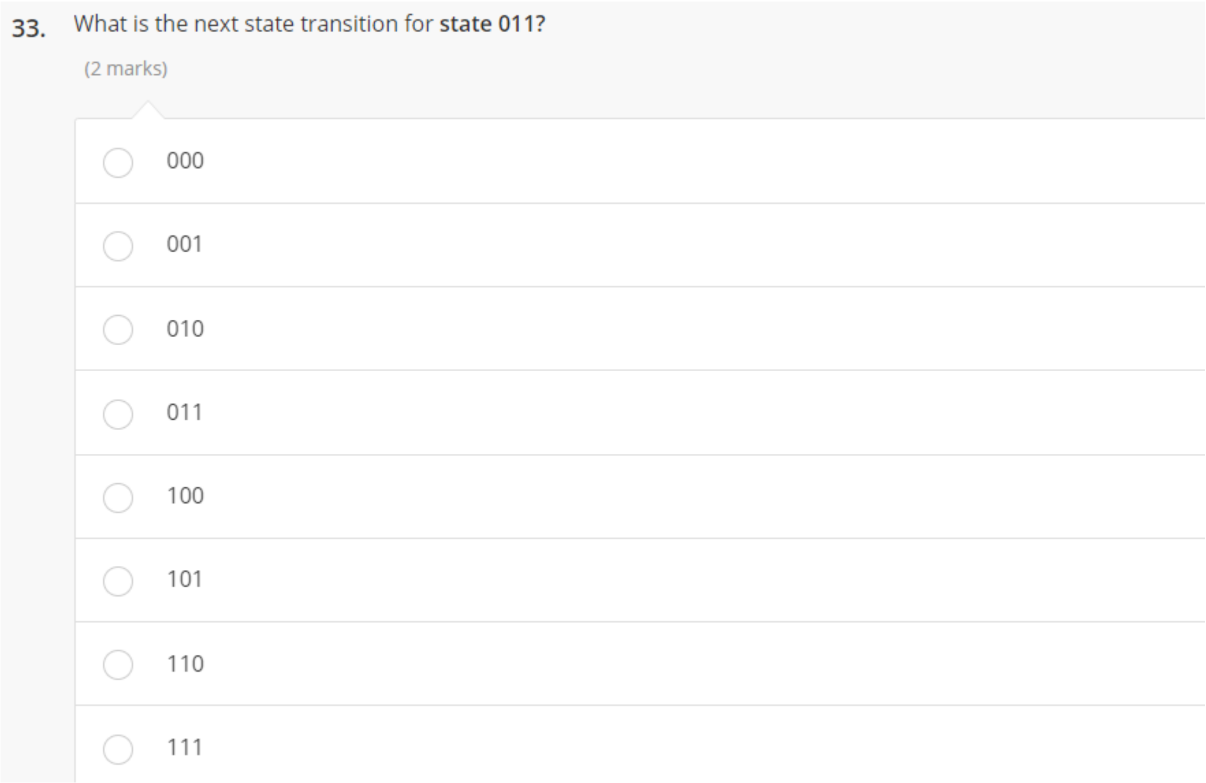
☐ F2

☐ F3



32. Give the simplifed SOP for **TC**.

(3 marks)



33. What is the next state transition for **state 011?**

(2 marks)

○ 000

○ 001

○ 010

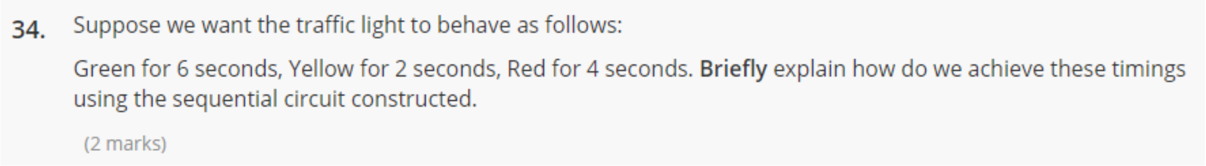
○ 011

○ 100

○ 101

○ 110

○ 111



34. Suppose we want the traffic light to behave as follows:

Green for 6 seconds, Yellow for 2 seconds, Red for 4 seconds. **Briefly** explain how do we achieve these timings using the sequential circuit constructed.

(2 marks)

~~~~ End of Paper ~~~
~~~~