CS2100 Computer Organisation
AY2021/22 Semester 1
Assignment 3

PLEASE READ THE FOLLOWING INSTRUCTIONS VERY CAREFULLY.

1. The **deadline** for this assignment is **8 November 2021, Monday, 1pm**. The submission folders will close shortly after 1pm and no submissions will be allowed after that. You WILL receive 0 mark if you do not submit on time.
2. You should complete this assignment **on your own** without collaborating or discussing with anyone. If you have been found to have cheated in this assignment, you will receive an F grade for this module as well as face other disciplinary sanctions. Take this warning seriously.
3. Please submit only **ONE pdf file** called **AxxxxxxxY.pdf**, where **AxxxxxxxY** is your student number. Please remove any old assignment files in the folder. Your folder should have only one file, or one mark will be deducted.
4. Please keep your submission **short**. You only need to submit your answers; you do not need to include the questions.
5. Answers may be typed or handwritten. In case of the latter, please ensure that you use **dark ink** and your handwriting is **neat and legible**. Marks may be deducted for untidy work or illegible handwriting.
6. Upload your pdf file to LumiNUS > Files > Assignment 3 > (your tutorial group) > (your personal folder).
7. There are THREE (3) questions (excluding question 0) on FIVE (5) printed pages in this paper..
8. The questions are worth **40 marks** in total.
9. If you have any queries on this assignment, please post them on LumiNUS > Forum > Assignments. Queries posted elsewhere will not be answered.

9. You do not need to show your working in this assignment.

=== END OF INSTRUCTIONS ===

Question 0. Submission instructions (3 marks)

You do not need to answer this question. Your tutor will award the marks accordingly.

(a) You have named your file with your student number, i.e., **AxxxxxxxY.pdf**. [1 mark]

(b) You have submitted your assignment as a **single PDF file** and no multiple copies, i.e. there should only be ONE file in your submission folder. [1 mark]

(c) Your submission has your **tutorial group number**, **student number** and **name** at the top of the first page of your file. (All three items must be present.) [1 mark]

Question 1. MSI Devices (16 marks)

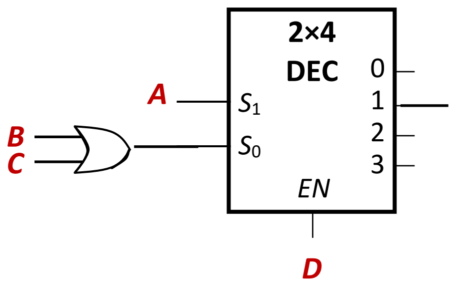
Note:

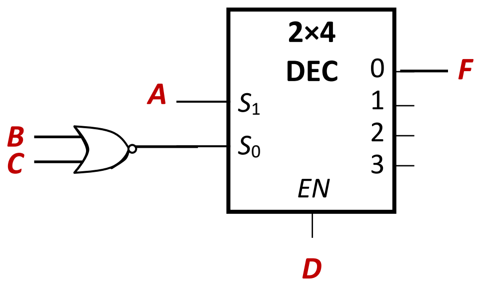
- Boolean constants (0 and 1) are always available.
- Complemented literals are not available.

(a) Implement the following Boolean function using a single 2×4 decoder with one-enable and active high outputs, and at most one logic gate. Logic gates available are NOT, OR, AND, NOR, NAND, XOR and XNOR. [4 marks]

$$F(A,B,C,D) = \Sigma m(3, 5, 7)$$

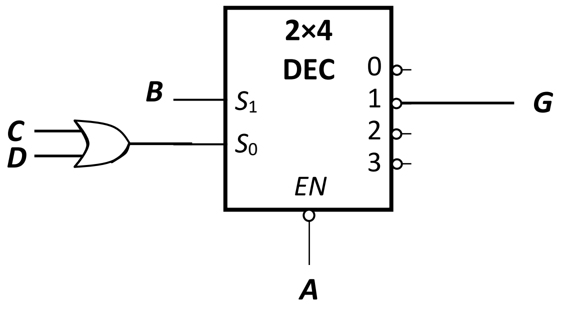
The block diagram of a 2×4 decoder with one-enable and active high outputs is shown below.





Other answers possible, as long as it gives $F = A' \cdot D \cdot (B+C)$.

(b) Given the circuit below implemented using a 2×4 decoder with zero-enable and active low outputs, write out the simplified POS (product-of-sums) expression for *G*. [4 marks]



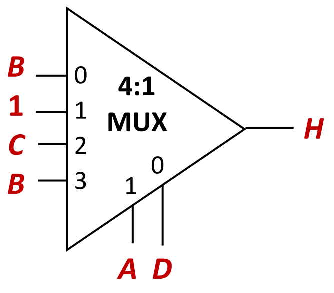
Answer: $G = (A + B + C') \cdot (A + B + D')$

(c) Implement the following Boolean function using a single 4:1 multiplexer without any logic gate. Once you have chosen the 2 variables for the selector lines, those 2 variables are not to appear in the 4 multiplexer inputs.

$$H(A,B,C,D) = \Sigma m(1, 3, 4, 5, 6, 7, 10, 13, 14, 15)$$

No mark will be awarded if any logic gate besides the multiplexer is used. The block diagram of a 4:1 multiplexer is given below. [4 marks]

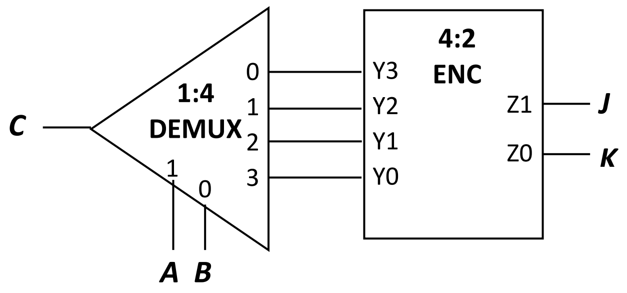
Answer:



A	*B*	*C*	*D*	*H*
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1

A	*B*	*C*	*D*	*H*
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

(d) The circuit below is implemented using a 1:4 demultiplexer and a 4:2 priority encoder. The function table of the 4:2 priority encoder is given below.



Inputs				Outputs	
Y3	Y2	Y1	Y0	Z1	Z0
0	0	0	0	X	X
0	0	0	1	0	0
0	0	1	X	0	1
0	1	X	X	1	0
1	X	X	X	1	1

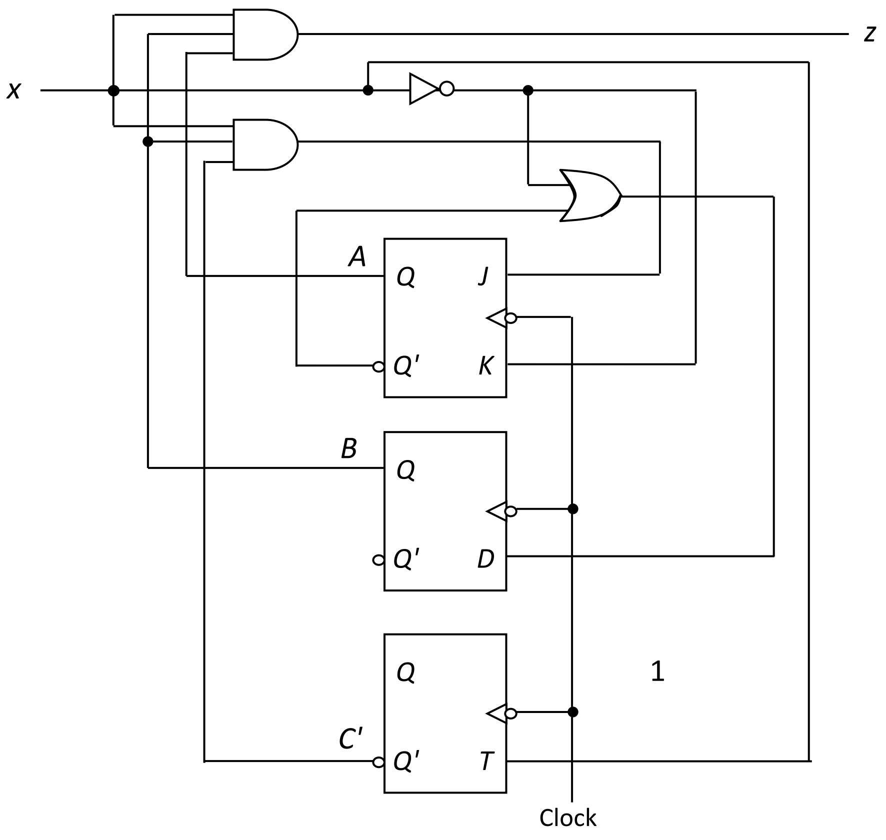
Write out the simplified SOP expressions for *J* and *K*. [4 marks]

Answer: *J* = *A*'; *K* = *B'*

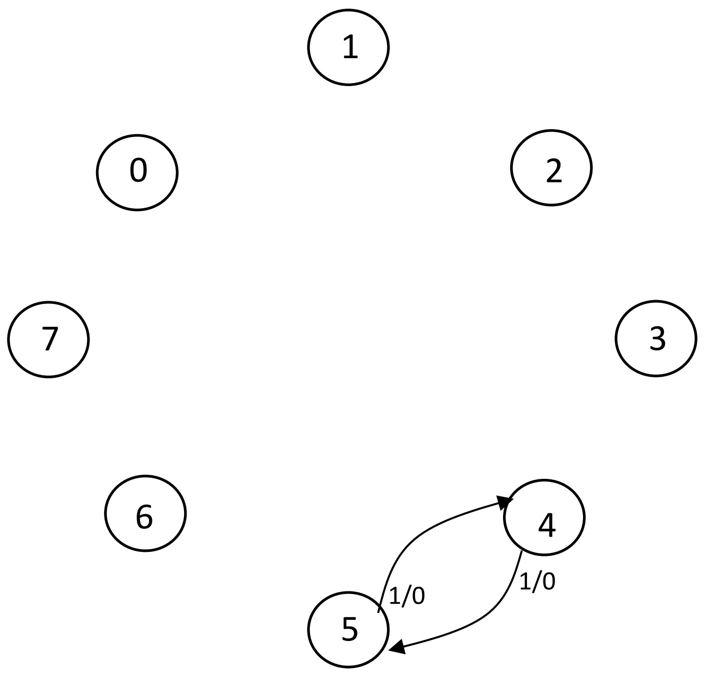
A	*B*	*C*	*Y3*	*Y2*	*Y1*	*Y0*	*J*	*K*
0	0	0	0	0	0	0	X	X
0	0	1	1	0	0	0	1	1
0	1	0	0	0	0	0	X	X
0	1	1	0	1	0	0	1	0
1	0	0	0	0	0	0	X	X
1	0	1	0	0	1	0	0	1
1	1	0	0	0	0	0	X	X
1	1	1	0	0	0	1	0	0

Question 2. Sequential Circuit (14 marks)

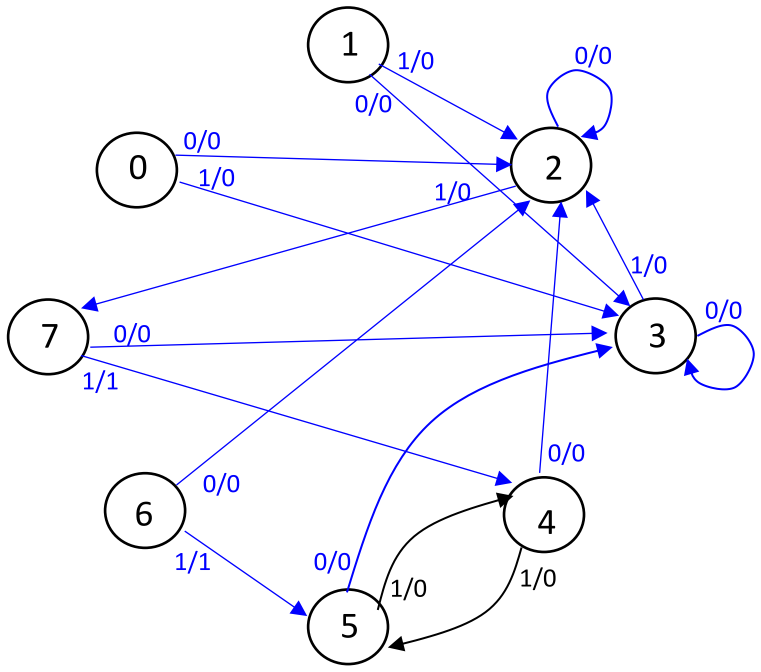
Analyze the following sequential circuit with JK flip-flops, an input *x* and an output *z*. The states are represented by *ABC*.



Complete the state diagram below. States are shown in decimal and the arrow labels are in *x*/*z* format, where *x* is the input and *z* the output. Two arrows have been drawn for you. For ease of tracing, write the label <u>near the tail of its associated arrow</u> (see the examples below). Use the same positions of the states in the diagram below to ease grading. You do NOT need to show your state table.



Answer:



State table:

Present state			Input	Output	Flip-flop *A*		Flip-flop *B*	Flip-flop *C*	Next state		
A	*B*	*C*	*x*	*z*=*x*·*A*·*B*	*JA*=*x*·*B*·*C*′	*KA*=*x*′	*DB*=*x*′+*A*′	*TC*=*x*	A^+	B^+	C^+
0	0	0	0	0	0	1	1	0	0	1	0
0	0	0	1	0	0	0	1	1	0	1	1
0	0	1	0	0	0	1	1	0	0	1	1
0	0	1	1	0	0	0	1	1	0	1	0
0	1	0	0	0	0	1	1	0	0	1	0
0	1	0	1	0	1	0	1	1	1	1	1
0	1	1	0	0	0	1	1	0	0	1	1
0	1	1	1	0	0	0	1	1	0	1	0
1	0	0	0	0	0	1	1	0	0	1	0
1	0	0	1	0	0	0	0	1	1	0	1
1	0	1	0	0	0	1	1	0	0	1	1
1	0	1	1	0	0	0	0	1	1	0	0
1	1	0	0	0	0	1	1	0	0	1	0
1	1	0	1	1	1	0	0	1	1	0	1
1	1	1	0	0	0	1	1	0	0	1	1
1	1	1	1	1	0	0	0	1	1	0	0

Question 3. Pipelining (7 marks)

The following MIPS code reads a 10-element integer array ***A***, whose base address is stored in **$s0**, to determine if the array is sorted in (non-strictly) increasing order. If it is sorted (example: -9, -3, 2, 2, 5, 8, 10, 10, 12, 15), **$s1** is assigned 1 (true), otherwise, **$s1** is assigned 0 (false).

```
      addi $s1, $zero, 1     # Inst1: sorted = true
      addi $s2, $s0, 36      # Inst2: $s2 = addr of A[9]
      addi $t0, $s0, 0       # Inst3
Loop: beq  $t0, $s2, Out     # Inst4
      lw   $t1, 0($t0)       # Inst5: $t1 = A[i]
      lw   $t2, 4($t0)       # Inst6: $t2 = A[i+1]
      slt  $t3, $t2, $t1     # Inst7
      beq  $t3, $zero, Next  # Inst8
      addi $s1, $zero, 0     # Inst9: sorted = false
      j    Out               # Inst10
Next: addi $t0, $t0, 4       # Inst11
      j    Loop              # Inst12
Out:
```

For parts (a) to (d) below, you are to consider only the first iteration of the above code in a 5-stage MIPS pipeline, that is, instruction 1 through instruction 12. You need to count until the last stage (WB stage) of instruction 12. You may assume that array ***A*** is sorted in (non-strictly) increasing order.

There is no partial credit given for each part. There is no need to show the pipeline chart (it will not be graded), although showing it may help us spot your errors.

(a) In an ideal pipeline with no delays, how many cycles are needed to complete the first iteration of the code? You do not need to show working or give explanation. [1 mark]

For parts (b) – (d) below, list out the cycles of delay at each instruction, and the total cycles of delay for the first iteration, that is, instruction 1 through instruction 12.

For example, for part (b), if you think that two cycles of delays are incurred at Instruction 4, one cycle of delay at Instruction 7, and three cycles of delays at Instruction 11, then you write:

Inst4:	+2
Inst7:	+1
Inst11:	+3
Total:	+6

(b) Assume without forwarding and branch is resolved at stage 4 (MEM). No branch prediction is made and no delayed branching is used. [2 marks]

(c) Assume with forwarding and branch is resolved at stage 4 (MEM). No branch prediction is made and no delayed branching is used. [2 marks]

(d) Assume with forwarding and branch is resolved at stage 2 (ID). No branch prediction is made and no delayed branching is used. [2 marks]

Answers:

(a) 10 instructions are executed (Instructions 1-8, 11 and 12).
Number of cycles = 10 + 5 – 1 = **14**. (1 mark)

(b) Number of cycles of delay = **12**. (2 marks)

Inst4:	+2
Inst5:	+3
Inst7:	+2
Inst8:	+2
Inst11:	+3
Total:	+12

Inst	*Cycle*																	
	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*	*16*	*17*	*18*
1: addi	F	D	E	M	W													
2: addi		F	D	E	M	W												
3: addi			F	D	E	M	W											
4: beq				F	-	-	D	E	M	W								
5: lw										F	D	E	M	W				
6: lw											F	D	E	M	W			
7: slt												F	-	-	D	E	M	W
8: beq													F	-	-	-	-	D

Inst	*Cycle*												
	18	*19*	*20*	*21*	*22*	*23*	*24*	*25*	*26*	*27*	*28*	*29*	*30*
8: beq	D	E	M	W									
11: addi				F	D	E	M	W					
12: j					F	D	E	M	W				

(c) Number of cycles of delay = **7**. (2 marks)

Inst5: +3
Inst7: +1
Inst11: +3
Total: +7

Inst				*Cycle*																	
	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*	*16*	*17*	*18*	*19*	*20*	*21*
1: addi	F	D	E	M	W																
2: addi		F	D	E	M	W															
3: addi			F	D	E	M	W														
4: beq				F	D	E	M	W													
5: lw								F	D	W	M	W									
6: lw									F	D	E	M	W								
7: slt										F	D	-	E	M	W						
8: beq											F	D	-	E	M	W					
11: addi																F	D	E	M	W	
12: j																	F	D	E	M	W

(d) Number of cycles of delay = **5**. (2 marks)

Inst4: +1
Inst5: +1
Inst7: +1
Inst8: +1
Inst11: +1
Total: +5

Inst			*Cycle*																	
	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*	*16*	*17*	*18*	*19*	*20*
1: addi	F	D	E	M	W															
2: addi		F	D	E	M	W														
3: addi			F	D	E	M	W													
4: beq				F	-	D	E	M	W											
5: lw							F	D	E	M	W									
6: lw								F	D	E	M	W								
7: slt									F	D	-	E	M	W						
8: beq										F	-	-	D	E	M	W				
11: addi														F	D	E	M	W		
12: j															F	D	E	M	W	

=== END OF PAPER ===