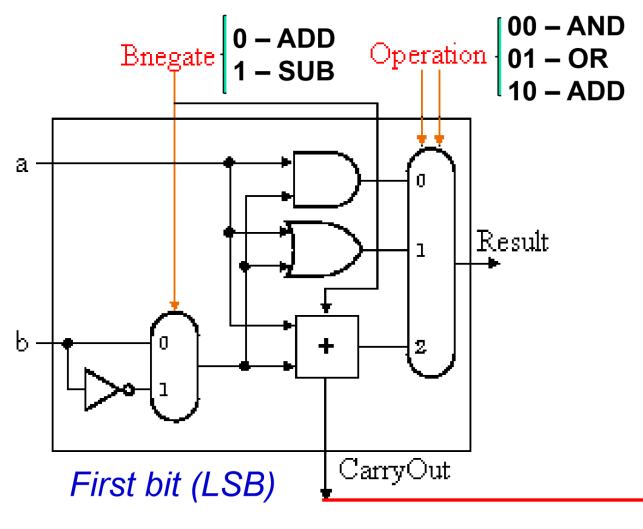


**ADD operation**

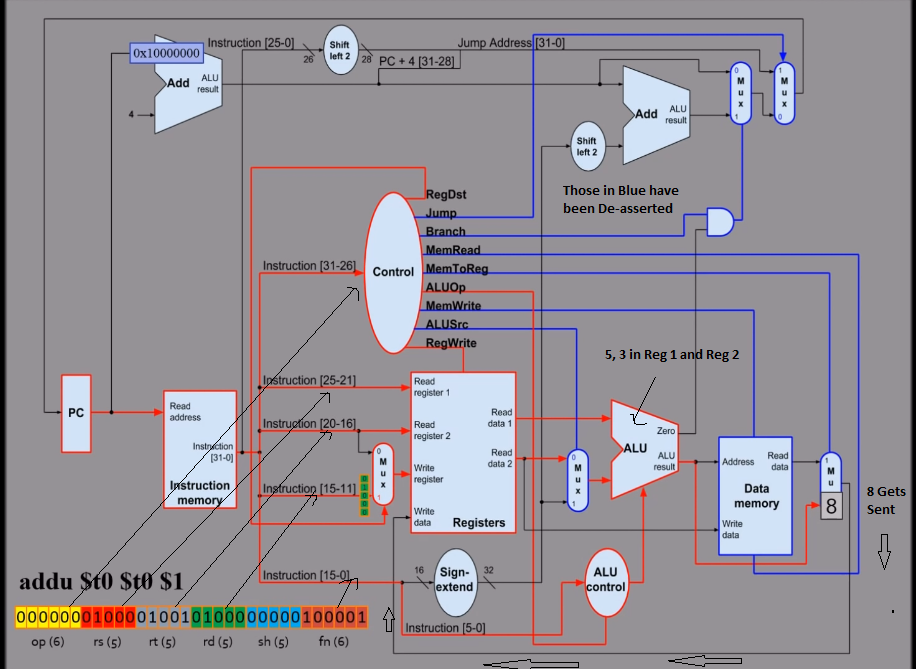
****

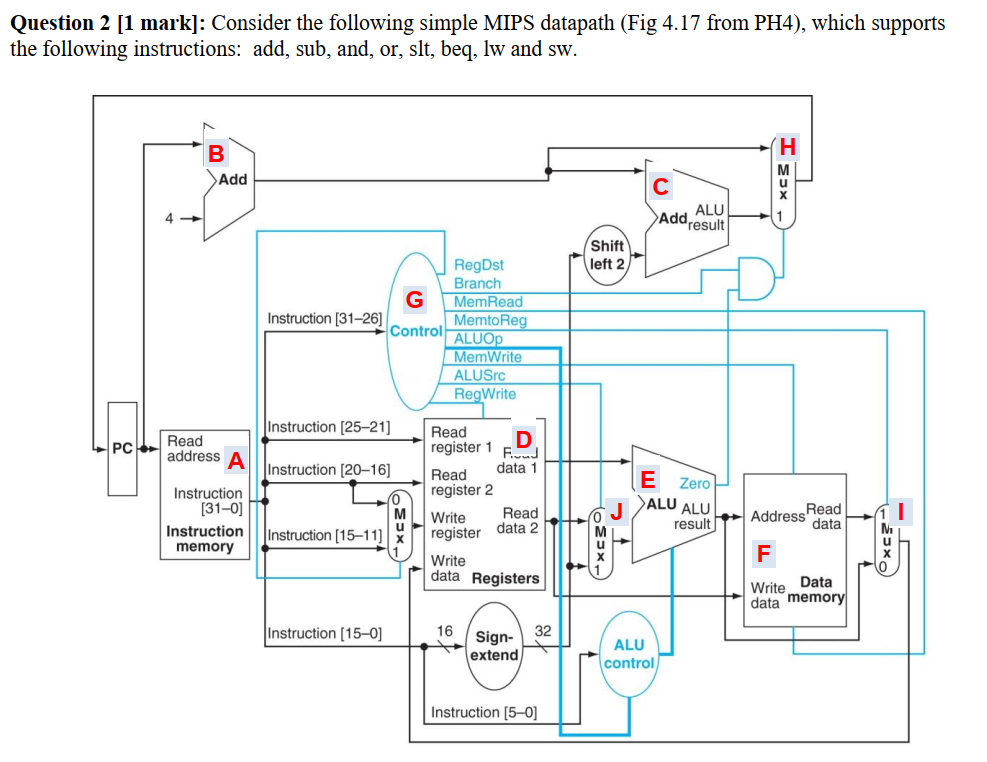
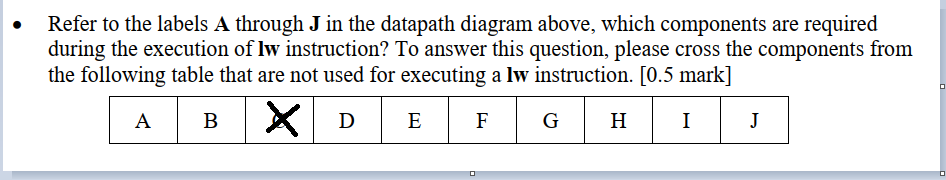
1. Program Counter is used to Fetch Instruction at Memory Address
2. Instruction is fetched from this memory location
3. The instruction is broken into pieces [masks]
4. The operation field is sent to the control Unit [sets up which path will be used; de-asserts or asserts mux to determine information flow]
5. Read Register 1, 2, are determined in REGwrite
6. Write Register is placed in REGwrite
7. Function [last field] is sent to ALU control
8. ALU control sends signal to ALU to determine which operation to do (ADD; 10 )

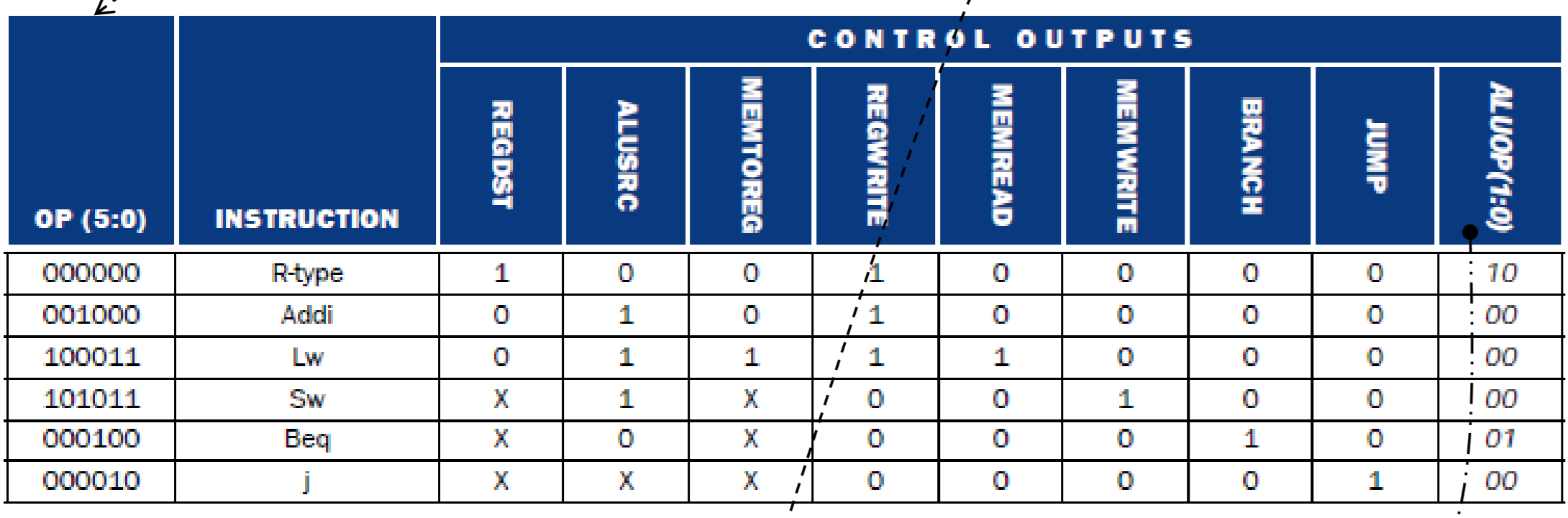


NOTE: ALU OP sends information to ALU control to tell ALU which data it is going to use

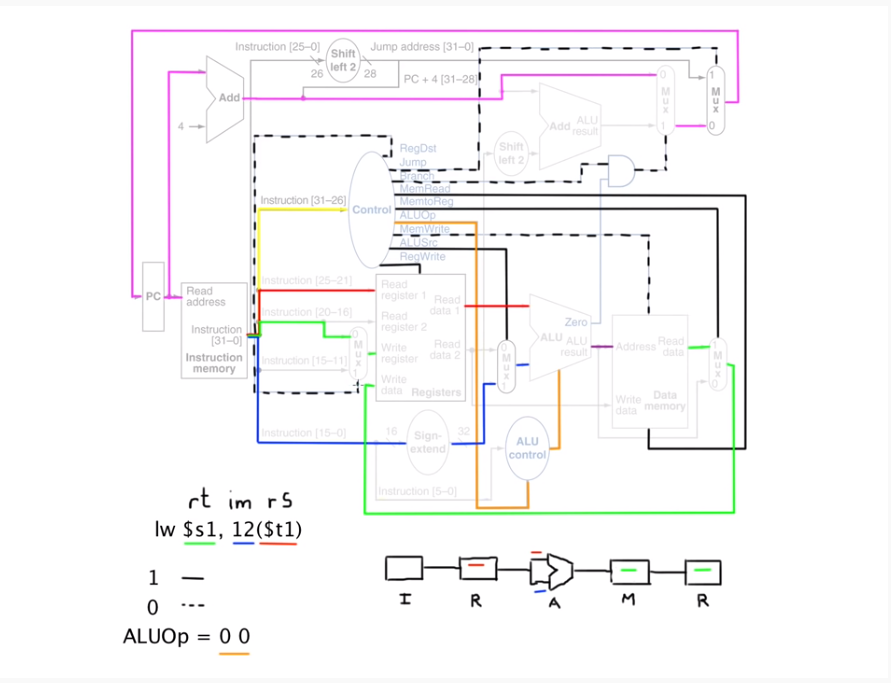
In this case: The ALU control knows that JUMP, Branch, and memory operations are going to be set to Zero (since this is an R-type instruction).



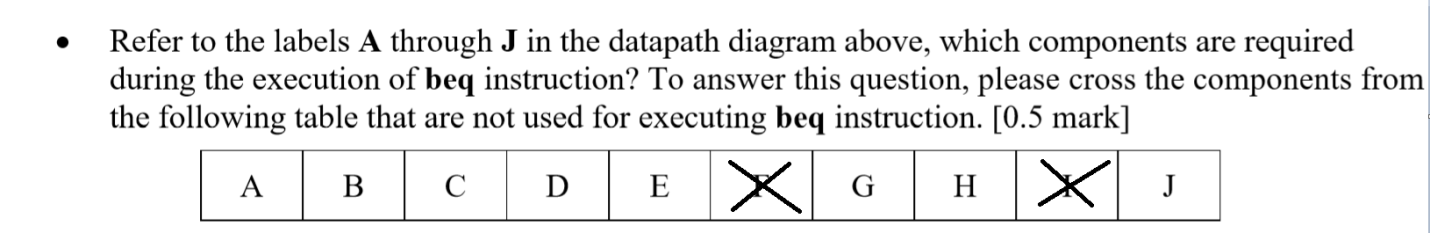
  


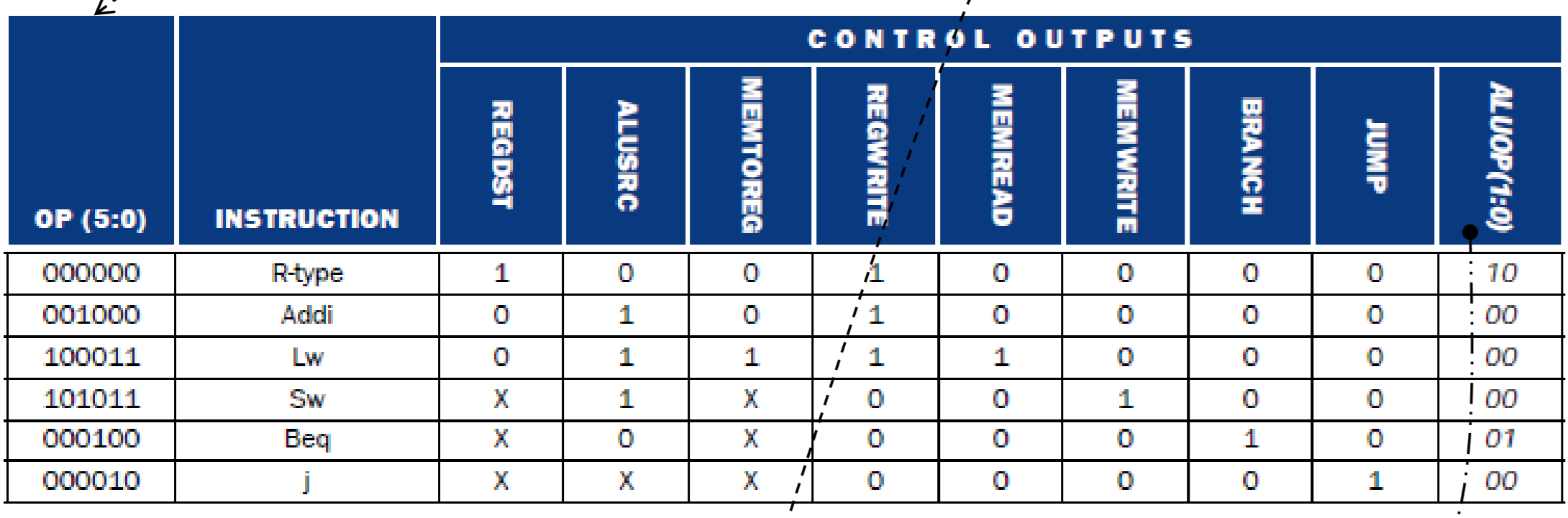


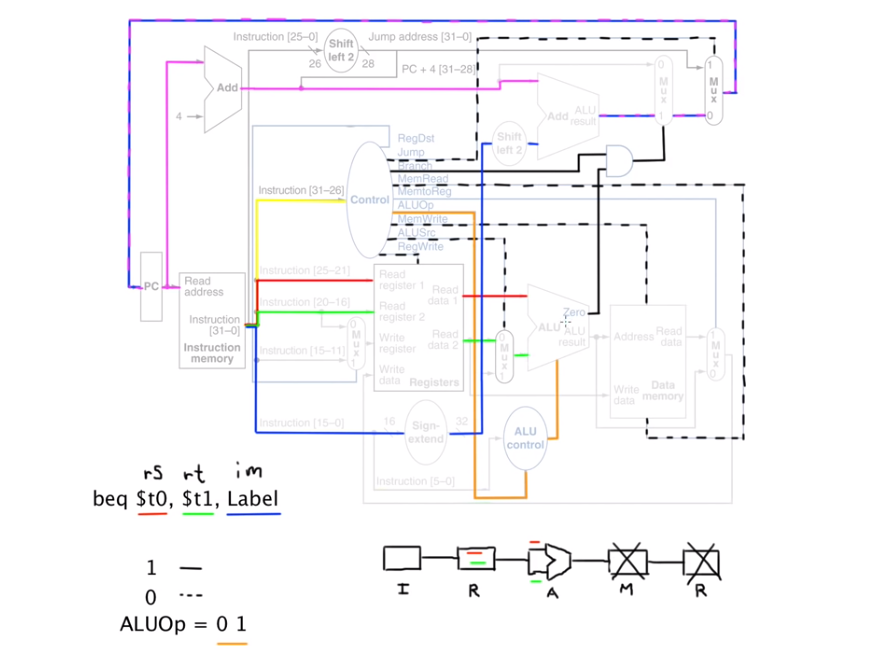
1. Instruction is Fetched [A,B,H]
2. Instruction is Decoded[A]
3. Operation field Sent to Control Unit (G)
4. Control Unit Asserts or De-asserts to control data flow; ALUSRC[J]
5. Offset is added to $t1 [E]
6. Write Data to register[F, D]
7. Memory to Register is set to 1 [I]



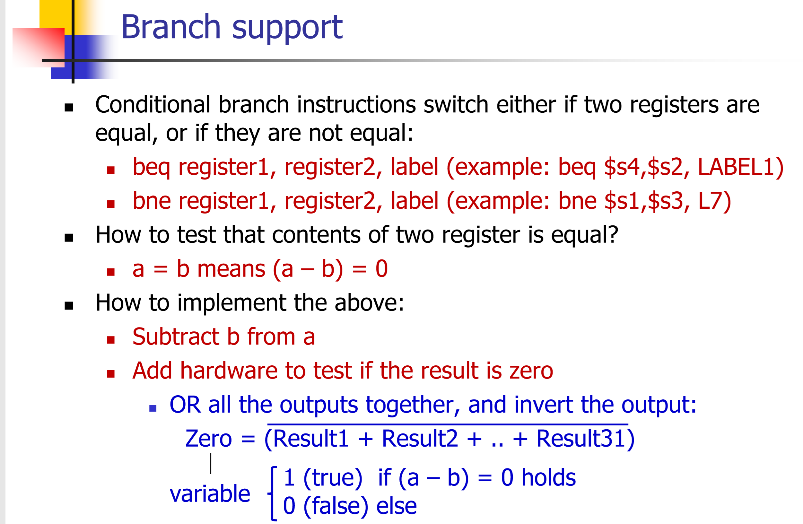
1. Green Label is to write to Particular address
2. $t1 sent to ALU
3. ALUoperation to add offset plus register $t1
4. Mem-Read Asserted [memory to register]

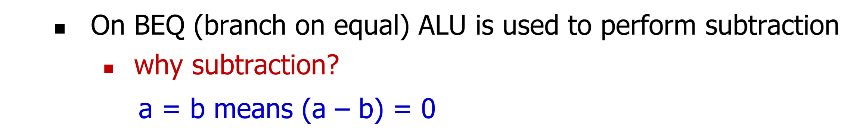


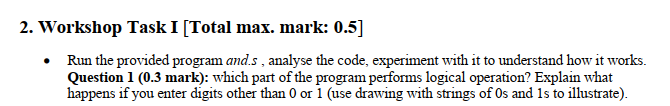




1. Fetch Instruction
2. Decode Instruction
3. ALU control
4. Branch
5. ALUop
6. ALU control allows $t1, $t0 tested [by minusing]
7. ALU asserts 1 at Branch AND value
8. If true, Added to program counter; to fetch new instruction



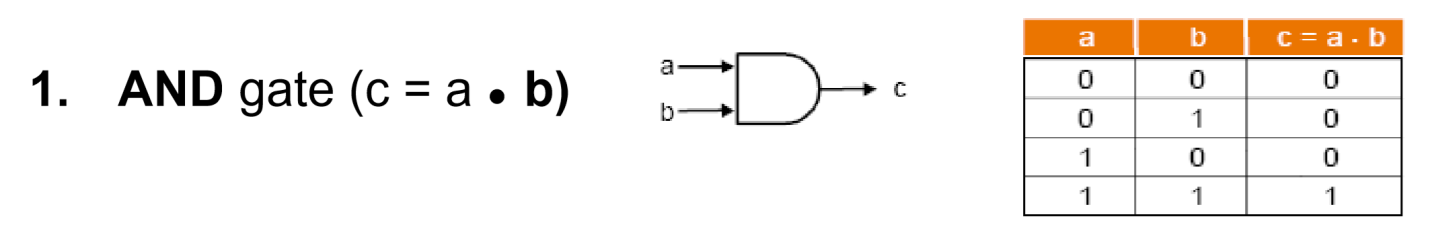




Takes two inputs; ANDs and Prints Results.  
  
Result is ANDED at line 41.  
  
  
  
  
  
All the ones which are in a common location will be remaning in the destination register. For instance:

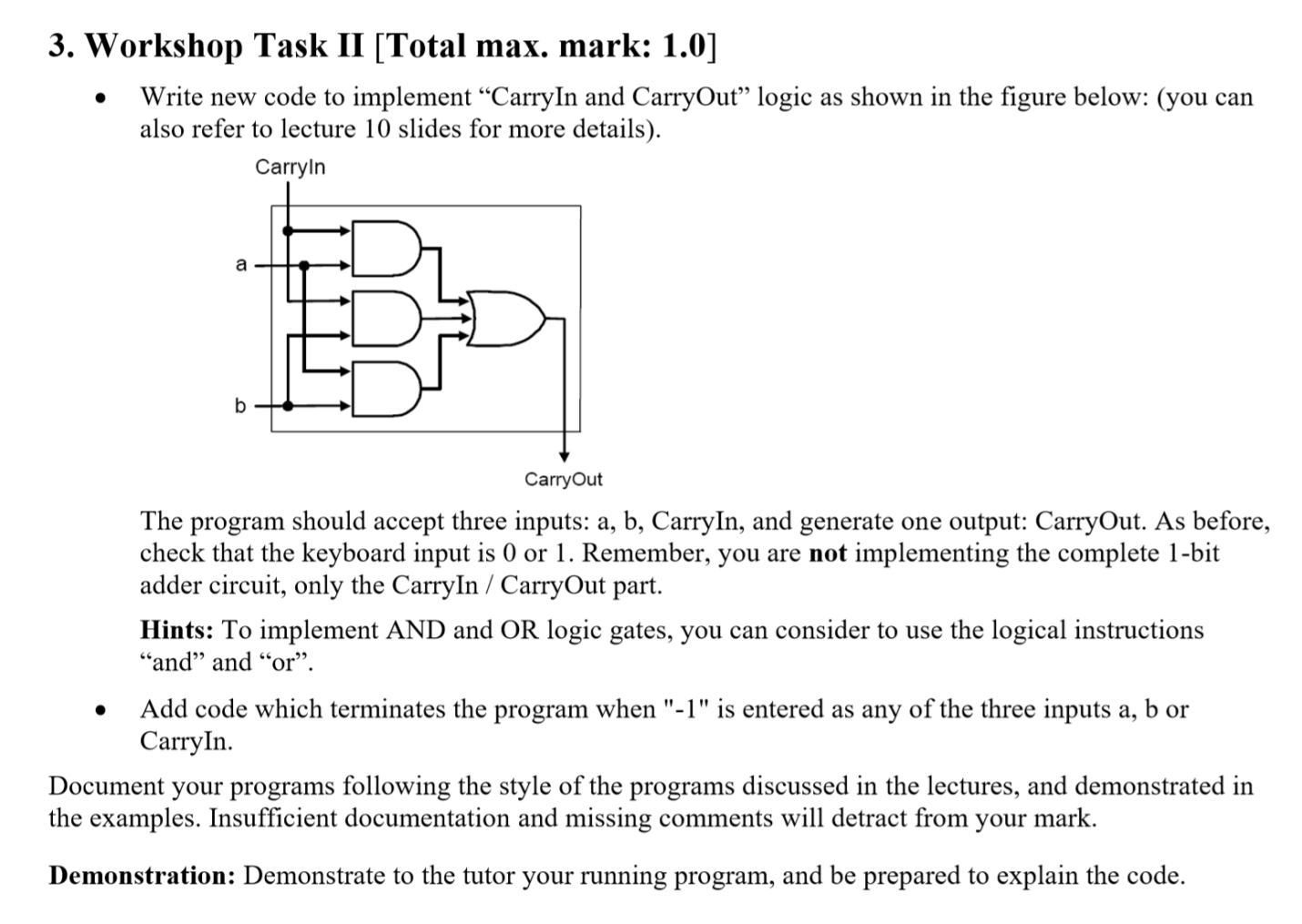
Register $s3 : 0001  
Register $s4 : 0000

Result(in $t0): 0000



Any value which is given to it other than one or zero will result in the program converting it to a binary digit. For instance; If I were to press “5” and “1”:  
  
Binary equivalent(5): 0101   
Binary equivalent(1): 0001  
  
AND operation(result): 0001

  
**Completed.**



**Completed.**

