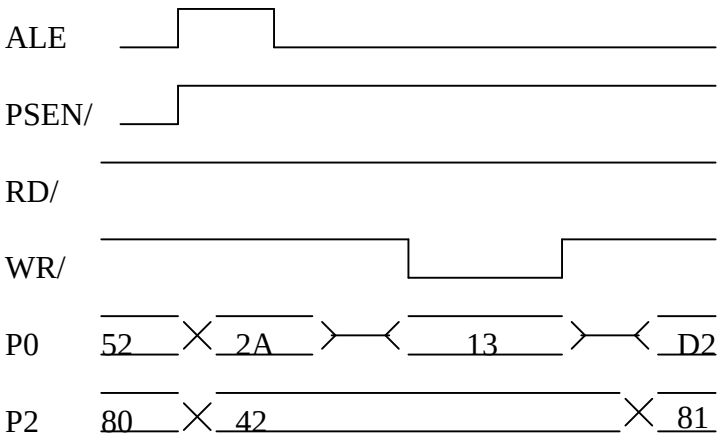
# CmpE213 – Test I

Name_____

Show all your work in the space provided. Answers with a simple "yes", "no", or a single number are typically incomplete and will not be given full credit. Answers in non-reduced form, like (a+sqrt(b))/c, are fine where appropriate. Good English on essay/short answer questions is required.

1. (18 Points) The following shows a short portion of a timing diagram from an 8051. For this diagram, find (and indicate how you got your results!):
   a) (6 Points) Whether we are reading/writing code or data,
   b) (6 Points) The address of the code or data accessed,
   c) (6 Points) The value of that data. If the data represents an instruction, find which instruction was read.

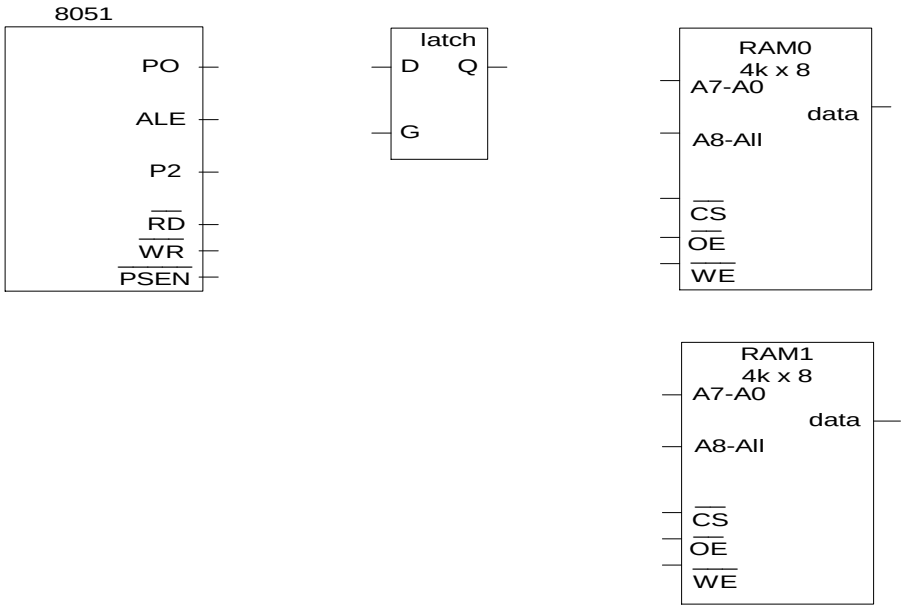



2. (15 Points) Explain why it is possible that after the following code sequence:

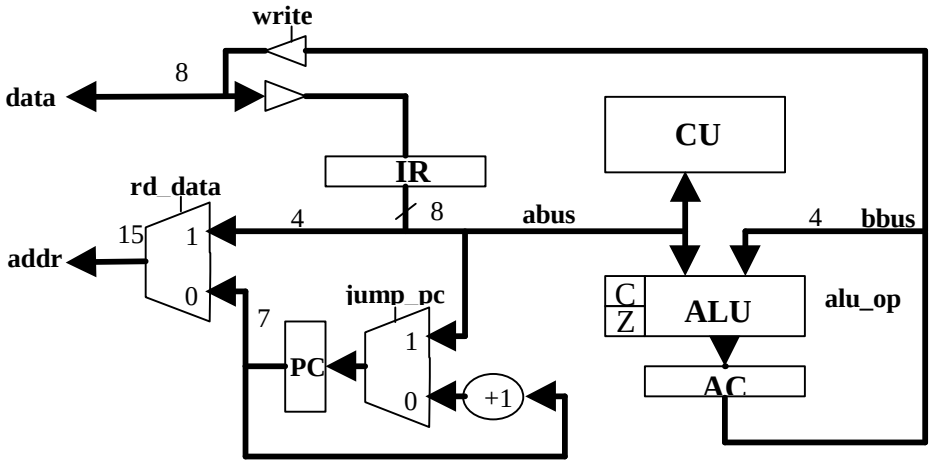
```
SETB P1.0
MOV C, P1.0
```

C may contain a zero. Be specific, alluding to any 8051 hardware that may allow this to happen (this is a hint). If we had written a 0 to P1.0 instead, would C ever contain a 1?

3. (16 Points) For the following diagram, complete the wiring of address, control, and data lines so the 8051 may communicate with the external RAM modules. Give RAM0 an address space of 0x8000-0xBFFF and RAM1 an address space of 0x4000-0x7FFF. (If you need to break individual lines from a multi-line bus, you may do so as shown on pg 43 of your text). Please label lines appropriately.




4. (18 Points) The 8051's command "MOV A,@R0" can be a very powerful command, as it is the basis for array access.
   a. (6 Points) Modify the GNOME architecture below to allow for a new GNOME instruction, "MOV A,@A" (i.e. A<- MEM(A)).
   b. (6 Points) Illustrate how this instruction would be performed by drawing arrows showing data-movement and explaining each step.
   c. (6 Points) Add an opcode to the GNOME instruction set for the instruction "MOV A,@A".



5. (33 Points) Answer the following questions for the given ASM code sequence and memory values for the 8051 (As a code sequence, instructions are executed IN ORDER).
   a. (15 Points) Show the values of any register or memory locations that are changed as this code-segment executes – please make it clear which changes take place with which instruction. Be sure to show changes to ANY SFR, including PC. Consider execution complete when the code can't progress any further.
   b. (6 Points) What is the total amount of code-space occupied by this code-sequence?
   c. (6 Points) How long did this code-sequence take to execute, assuming a *42MHz* clock? (note that we used a 12MHz clock in the homework).
   d. (6 Points) Find the opcode for instruction number 4 (ACALL blah).

| Addr. | Inst. # | Instruction | | # bytes | # machine cycles |
|---|---|---|---|---|---|
| C:0x0000 | 1. | | MOV 2AH, #A0H | 3 | 2 |
| C:0x0003 | 2. | | MOV @R0, #0DFH | 2 | 2 |
| C:0x0005 | 3. | | PUSH 2AH | 2 | 2 |
| C:0x0007 | 4. | | ACALL blah | 2 | |
| C:0x0009 | 5. | stop: | SJMP stop | 2 | 2 |
| C:0x000B | 6. | blah: | SETB 2AH | | |
| | 7. | | RET | 1 | 2 |

| inst.# | SFR's | | | | | | Internal Mem location | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PSW | R0 | SP | A | B | PC | 25H | 26H | 27H | 28H | 29H | 2AH | 2BH |
| | 97H | 29H | 25H | 42H | 13H | 0000H | 00H | 01H | 2AH | F7H | 00H | FFH | 24H |
| 1. | | | | | | 0003H | | | | | | A0H | |
| 2. | | | | | | | | | | | | | |
| ? | | | | | | | | | | | | | |