2. **[16 points] Grab Bag**

(a) Prof. Ross Esser wants to add a new subtract-immediate instruction to the LC-4 ISA. The instruction's assembly syntax would be `SUB Rd Rs IMM6`, and its semantics would be `Rd = Rs - sext(IMM6)`. Propose a way to encode this new instruction such that it does not conflict with any existing instructions, in the format used for the *Encoding* column for the LC4 ISA table.

[2 points] 0011 ddd sss iiiiii

(b) Given the Lab 5 superscalar pipeline, fill in a pipeline diagram for the instructions below, showing when each instruction reaches each stage and whether it is in the *a* pipeline or the *b* pipeline (see example in cycle 1). Use a * to indicate a stall (you don't have to distinguish different stall types, e.g., superscalar versus branch misprediction).

insn	1	2	3	4	5	6	7	8	9	10	11
`STR r0 -> r1 #4`	F_a	Da	Xa	Ma	Wa						
`LDR r0 <- r1 #5`	F_b	Db	Da	Xa	Ma	Wa					
`ADD r4 <- r2 r3`		Fa	Db	Xb	Mb	Wb					
`NOT r5 <- r0`		Fb	Fa	Da	*	Xa	Ma	Wa			

[10 points] 1pt for STR, 3pts each for other insns

(c) Dr. Carrie Luk, a head computer architect, asks you for the infinite-hardware CLA equation that computes a carry-out in terms of generate and propagate bits. Give a formula to compute the carry-out C_3 in terms of the initial carry-in C_0 and the various G_j and P_k terms.

[4 points] $C_3 = G_2 \vee (P_2 \wedge G_1) \vee (P_2 \wedge P_1 \wedge G_0) \vee (P_2 \wedge P_1 \wedge P_0 \wedge C_0)$

2. **[12 points] Branch Prediction, Superscalar**

(a) List the states (1-letter abbreviations are fine) that a 2-bit saturating counter can have, as used in a branch history table.

N, n, t, T

(b) Given the Lab 5 superscalar pipeline, and the contents below, fill in two insns in Decode that will result in **no** stall in pipe A and a **superscalar** stall in pipe B.

	Decode	**Execute**	**Memory**	**Writeback**
pipe A	ADD R1, R3, R7	ADD R1, R3, R2	LDR R4, R6, #1	STR R1, R2, #2
pipe B	LDR R7, [R3, 8]	CMP R2, R5	SUB R3, R3, R1	XOR R3, R1, R2

(c) Senator Amidalu is working on a new area-efficient ALU design for the Lab 5 superscalar processor. Her idea is to share the large DIV/MOD unit between the two pipes, saving area but only allowing one DIV or MOD insn in Execute at any one time. Complete the module below which detects (*only*) this new stall during Decode.

```
module divModStall(input wire [15:0] decodeInsnA,
                   input wire [15:0] decodeInsnB,
                   output wire isDMStall);




endmodule
```