Part I. Regular Questions (hand in as hardcopy) [5 points total]

1. Vocabulary: (terms you need to know to discuss the subject intelligently) – Define the following terms using 1-3 sentences (and a diagram, if needed): [1 point each]
   a. MIPS PC-Relative Addressing Mode
   b. Pointer Arithmetic
   c. Stack Frame
   d. The register $s_p$ – what is it, and what is it for?
   e. Pointer Dereferencing operator (in C and how implemented in MIPS)

Part II. MIPS Coding Problems (hand in as hardcopy) [35 points total]

2. Simple MIPS Coding: A C-like pseudocode fragment comprises a program description that has the following statement(s) – please translate these into MIPS code, assuming the base address of vector $a$ is in register $s0$: [5 pts total]

   int a[20], b # $a$ is an array, $b$ is initialized to 17
   while $b < 22$ do:
   { $a[int(b/2)] = b * 13 ; b = b + 1$ }

   Document (comment) your code fully, including register assignments, for full credit.

3. Simple MIPS Coding: A C-like pseudocode fragment comprises a program description that has the following statement(s) – please translate these into MIPS code, assuming the base address of vector $a$ is in register $s0$: [5 pts total]
int a[20], b    # a is an array, b is initialized to 1
do:
{ a[b^2] = b * 13 ; b = b + 1 }
while b < 5 ;

Verify your program with the SPIM simulator before you submit it on paper. Document (comment) your code fully, including register assignments, for full credit.

4. Iterative Algorithm Design and MIPS Coding: Write a MIPS program to iteratively compute up to 35 Grumpy numbers $G(i) = 2^{G(i-1)} + 3 \cdot G(i-2)$ and store them sequentially in an array $G$ whose base address in MIPS memory is stored in register $s0$. Also include a procedure to “trap” overflow errors -- if $G(i)$ is too big to fit into a 32-bit register, your program should stop. Base case: $G(0) = 1, G(1) = 2$.

Please write the MIPS code (fully commented) to compute $G(i), 2 \leq i \leq 34$, then verify it with the SPIM simulator before you submit it on paper. [10 pts]

5. Recursive Algorithm Design and MIPS Coding: Suppose you want to compute recursively the first $n$ Combower numbers $C(i) = 13 \cdot C(i-1) - 2^{C(i-2)} + 7^{C(i-3)}$, for $n > 3$. Base case: $C(0) = 0, C(1) = 2$, and $C(2) = 13$. Please write MIPS code to perform this task recursively. Your code should be structured as follows: [5 pts for a), 10 pts for b)]

a. Main Program (calling routine): Has the number $n$ stored in register $s1$, and the base address of the array $C$ stored in register $s0$, with a counter $i$ in $t1$. Main program calls the routine (specified below) Combower $(n)$.

b. Callee Routine calls Combower $(i-1)$ through Combower$(i-3)$, then updates value of $n$ appropriately. Then function Combower computes the $i^{th}$ Combower number, checks it for overflow, then stores it in memory. If overflow is detected, the program stops.

The Result should compute the first $n$ Combower numbers recursively and store them in an array of $n$ words in memory. Verify your program with the SPIM simulator before you submit it on paper. All code MUST be fully documented.

Part III. Extra Credit: (hand in as hardcopy) [20 points total]

6. Outer Product: The outer product or tensor product of vectors $a$ and $b$ is given by $c = a \otimes b$, where $c(i,j) = a(i) \cdot b(j)$. For example, if vectors $a = (1, 2)$ and $b = (3, 4)$, we have the tensor product of $a$ and $b$ as the two-dimensional matrix $c$ which is computed as

$$c = a \otimes b = \begin{bmatrix} 1 \cdot 3 & 1 \cdot 4 \\ 2 \cdot 3 & 2 \cdot 4 \end{bmatrix}.$$ 

The scanning of matrix $c$ in normal order produces vector $(c(1,1), c(1,2), c(2,1), c(2,2))$. 
Write a MIPS program to *iteratively* compute the outer product of two \( n \)-element vectors, where we assume that \( n \leq 10^4 \), for convenience. The arrays in the MIPS .data section should be named \( a \), \( b \), and \( c \) – with \( c \) being stored in memory as scanned in normal order.

**Important Note:** Use the SPIM simulator to determine whether or not you can generate and store the array in MIPS memory without causing memory overflow (e.g., if \( n = 10^4 \) is too large). Then modify your MIPS program to only generate the array \( c \) up to the maximum value of \( n \) or the maximum size of \( c \) that can be stored in memory.

*Verify your program with the SPIM simulator before you submit it on paper.*
*Document your code fully,* including register assignments, to get full credit.