Problem #1 (40 points) Consider the following C program.

```c
#include "io430.h"
#include "stdio.h"

int main(void) {
    // Stop watchdog timer to prevent time out reset
    WDTCTL = WDTPW + WDTHOLD;
    int x = 5;  // an integer x
    int *p_x;   // a pointer to int
    int y1;     // an integer y1 (uninitialized)
    long int y2, y3; // long integers y2, y3
    long int *p_y2;  // a pointer to long integer
    char mya[10] = "Hello!"; // character array
    char *p_mya;    // pointer to character

    p_x = &x;        // p_x points to x
    *p_x = 7;
    y1 = 10 + x;     // new value to y1
    y2 = -1;
    *p_y2 = y2 + 3;
    y3 = 10 + *p_y2;
    p_mya = mya;     // p_mya points to array mya
    p_mya = p_mya + 3;
    *p_mya = 'L';

    // display addresses and variables in terminal i/o
    printf("a.x=%x, x=%x\n", &x, x);
    printf("a.p_x=%x, p_x=%x\n", &p_x, p_x);
    printf("a.y1=%x, y1=%x\n", &y1, y1);
    printf("a.y2=%x, y2=%lx\n", &y2, y2);
    printf("a.y3=%x, y3=%lx\n", &y3, y3);
    printf("a.p_y2=%x, p_y2=%x\n", &p_y2, p_y2);
    printf("a.mya=%x, mya=%s\n", &mya, mya);
    printf("a.p_mya=%x, p_mya=%x\n", &p_mya, p_mya);
    return 0;
}
```
A. (20 points). Illustrate the content of the stack at the moment (i) before the statement in line 15 is executed and (ii) before the statement in line 27 is executed. Use the comments fields to indicate the individual variables.

<table>
<thead>
<tr>
<th></th>
<th>i</th>
<th></th>
<th>ii</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Orig. TOS, Memory[15:0] hex, Comments</td>
<td></td>
<td>Orig. TOS, Memory[15:0] hex, Comments</td>
<td></td>
</tr>
<tr>
<td>OTOS-2</td>
<td></td>
<td></td>
<td>OTOS-2</td>
<td></td>
</tr>
<tr>
<td>OTOS-4</td>
<td></td>
<td></td>
<td>OTOS-4</td>
<td></td>
</tr>
</tbody>
</table>
B. (20 points). For each statement from line 15 to line 25 show its assembly language implementation.

<table>
<thead>
<tr>
<th>Assembly code</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>p_x = &amp;x;</td>
<td></td>
</tr>
<tr>
<td>*p_x = 7;</td>
<td></td>
</tr>
<tr>
<td>*p_x = *p_x + 2;</td>
<td></td>
</tr>
<tr>
<td>y1 = 10 + x;</td>
<td></td>
</tr>
<tr>
<td>y2 = -1;</td>
<td></td>
</tr>
<tr>
<td>p_y2 = &amp;y2;</td>
<td></td>
</tr>
<tr>
<td>*p_y2 = y2 + 3;</td>
<td></td>
</tr>
<tr>
<td>y3 = 10 + *p_y2;</td>
<td></td>
</tr>
<tr>
<td>p_mya = mya;</td>
<td></td>
</tr>
<tr>
<td>p_mya = p_mya + 3;</td>
<td></td>
</tr>
<tr>
<td>*p_mya = 'L';</td>
<td></td>
</tr>
</tbody>
</table>
2. (30 points) Consider the following assembly program.

```
#include "msp430.h" ; #define controlled include file
NAME main ; module name
PUBLIC main ; make the main label visible
; outside this module
ORG 0FFFEh ; set reset vector to 'init' label
RSEG CSTACK ; pre-declaration of segment
RSEG CODE ; place program in 'CODE' segment

init:  MOV #SFE(CSTACK), SP ; set up stack
main:  NOP ; main program
       MOV.W #WDTPW+WDTHOLD,&WDTCTL ; Stop watchdog timer
       PUSH R14
       MOV SP, R14
       SUB #0x2, SP
       MOV #arr1, R5
       CLR.B R7
       CLR.B R8
lnext: ADD.B @R5+, R7
       CMP #aend, R5
       JZ lexit
       ADD.B @R5+, R8
       CMP #aend, R5
       JZ lexit
       JMP lnext
lexit: MOV.B R7, -2(R14)
       MOV.B R8, -1(R14)
       JMP $
arr1 DC8 1, 2, 3, 4, 5, 6, 7, 8, 9
aend
END
```

A. (15 points) What does this program do?

B. (10 points) How many clock cycles does this program take to execute (do not consider the last instruction, JMP $). Show your work. What is the MIPS rate assuming that the clock frequency is 4 MHz?

C. (5 points) Sketch the content of the stack at the moment when the program executes the instruction at line 28. Assume that the original value of R14=0xEEEE.
3. **(30 points)** Design and write an MSP430 assembly language subroutine that returns the maximum absolute value of a signed integer array, i.e., $AMAX=\max(|a(i)|)$, $i=0, \ldots, n-1$, $n>0$. The main program that calls the subroutine is shown below:

```asm
#include "msp430.h" ; #define controlled include file
NAME main ; module name
PUBLIC main ; make the main label visible
EXTERN absmax ; outside this module
ORG 0FFFEh
DC16 init ; set reset vector to 'init' label
RSEG CSTACK ; pre-declaration of segment
RSEG CODE ; place program in 'CODE' segment
init: MOV #SFE(CSTACK), SP ; set up stack
main: MOV.W #WDTPW+WDTHOLD,&WDTCTL ; Stop watchdog timer
       BIS.B #0xFF,&P1DIR ; configure P1.x as output
       BIS.B #0xFF,&P2DIR ; configure P2.x as output
       PUSH #arr1 ; push the address of arr1
       MOV #aend, R5 ; the next address
       SUB #arr1, R5 ; (#aend - #arr1)/2 is the number of elements
       RRA R5 ; push the number of elements on the stack
       CALL #absmax
       ADD #4,SP ; collapse the stack ; display the maximum on P1OUT&P2OUT
       MOV.B R12, P1OUT ; display lower byte on P1OUT
       SWPB R12
       MOV.B R12, P2OUT ; display higher byte on P2OUT
       JMP $
arr1 DC16 1, 2, 3, 4, -5, -9, -12, 11 ; signed array
aend
END

PUBLIC absmax
RSEG CODE
absmax: ; write your code here
```