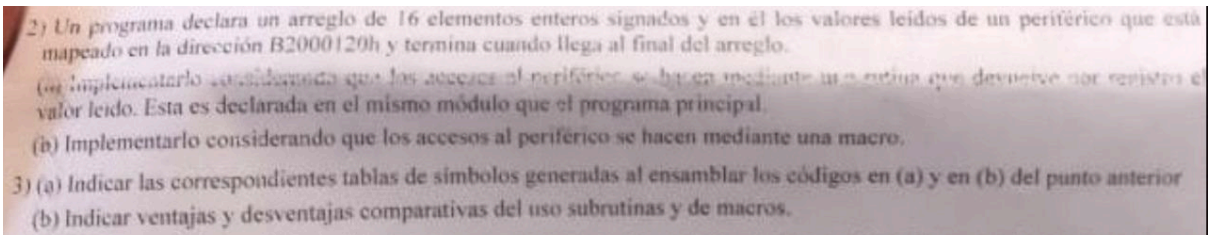
2) Un programa declara un arreglo de 16 elementos enteros signados y en él los valores leídos de un periférico que está mapeado en la dirección B2000120h y termina cuando llega al final del arreglo.
(a) Implementarlo considerando que los accesos al periférico se hacen mediante una rutina que devuelve por registro el valor leído. Esta es declarada en el mismo módulo que el programa principal.
(b) Implementarlo considerando que los accesos al periférico se hacen mediante una macro.
3) (a) Indicar las correspondientes tablas de símbolos generadas al ensamblar los códigos en (a) y en (b) del punto anterior
(b) Indicar ventajas y desventajas comparativas del uso subrutinas y de macros.

2. Un programa declara un array de 16 elementos enteros signados y en él los valores leídos de un periférico mapeado en la dirección B2000120 y termina cuando llega al final del array.

a. Implementarlo considerando que los accesos al periférico se hacen mediante una rutina que devuelve por registros el valor leído. Declarada en el mismo módulo que el programa principal.

```
                .begin
                .org 2048
array:          .dwb 16
largo:          .equ 64
main:           add largo, %r0, %r1
                add %r15, %r0, %r31
                call loop
loop:           andcc %r1, %r1, %r0
                be fin
                call lectura
                add %r1, -4, %r1
                st %r2, %r1, [array]
                ba loop
lectura:        ld [periferico], %r3
                st %r3, %r2
                jmpl %r15+4, %r0
periferico:     0xALGO
                .end
```

b. Implementando que los accesos al periférico se hacen en una macro.

```
                .begin
                .org 2048
                .macro lectura dirección
                st dirección, %r2
                .end macro
array:          .dwb 16
largo:          .equ 64
main:           ld[perif], %r6
                add largo, %r0, %r1
                call loop
```

```
loop:     andcc %r1, %r1, %r0
          be fin
          lectura %r6
          add %r1, -4, %r1
          st %r2, %r1, [array]
          ba loop
fin:      jmpl %r15+4, %r0
perif:    0xALGO
          .end
```

3.

a. Tablas de símbolos

2) a.

```
          .begin
          .org 2048
array:    .dwb 16                  2048
largo:    .equ 64
main:     add largo, %r0, %r1      2112
          add %r15, %r0, %r31      2116
          call loop                2120
loop:     andcc %r1, %r1, %r0      2124
          be fin                   2128
          call lectura             2132
          add %r1, -4, %r1         2136
          st %r2, %r1, [array]     2140
          ba loop                  2142
lectura:  ld [periferico], %r3     2146
          st %r3, %r2              2150
          jmpl %r15+4, %r0         2154
fin:      jmpl %r31+4, %r0         2158
periferico: 0x ALGO                2162
          .end
```

Símbolo	Valor
array	2048
largo	64
main	2112
loop	2124
lectura	2146

periférico	2162
fin	2158

Idem con el 2. b.

b. Ventajas y desventajas del uso de subrutinas y el uso de macros.

Recordemos que una macro es una porción de código que luego se reemplaza por lo declarado en la misma juntos con los argumentos en tiempo de ensamblado y que las subrutinas son porciones de código donde al momento de llamarlas se les pasa el control del programa.

Como ventaja de la macro podemos decir que es una buena forma de no repetir código, por ejemplo cuando hacemos las macros de push y pop de la pila.
Además utilizar macros y no subrutinas ahorra símbolos que tendrían que cargarse en memoria.

Con respecto a las subrutinas, nos son de gran utilidad para poder organizar el código y aumentar la legibilidad del mismo, lo que tienen de desventaja es que al momento de la ejecución, los saltos que se producen pueden afectar al tiempo de ejecución del programa.