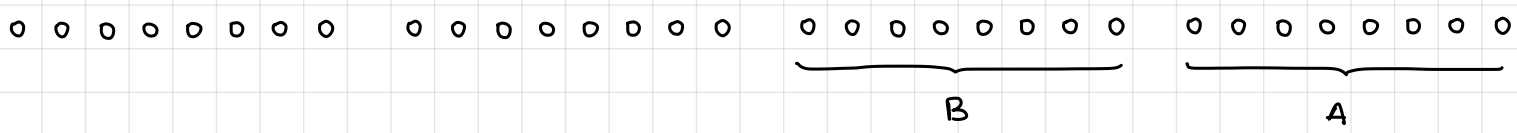
MODELO

1-Se reciben dos números signados de 8 bits por el registro %r7, A=A7..A0 y B= B15.. B8. Escribir un programa de ensamblador que compare A y B de manera que retorne 1 si A>B, 0 si son iguales y -1 si A<B.




```
.begin
   .org 2048
   sll  %r7, 24, %r10      ! r10 ← A A A A   0 0 0 0   0 0 0 0   0 0 0 0
   sra  %r10, 24, %r10     ! r10 ← x x x x   x x x x   x x x x   A A A A
   sll  %r7, 16, %r11      ! r11 ← B B B B   A A A A   0 0 0 0   0 0 0 0
   sra  %r11, 24, %r11     ! r11 ← x x x x   x x x x   x x x x   B B B B
   subcc %r10, %r11, %r0   ! r0 ← A - B
   bpos  uno               ! A > B
   bneg  menos_uno         ! A < B
   be    cero              ! A == B
   jmpl  %r15, 4, %r0      ! vuelvo
uno:  addcc %r0, 1, %r3    ! r3 ← 0+1
menos_uno:  addcc %r0, -1, %r3   ! r3 ← 0-1
cero:  st %r3, %r0         ! r3 = r0
.end
```

2-Lo/la contratan de una empresa para desarrollar un compilador C para procesadores de 32 bits. Realice una buena implementación en ensamblador para las sentencias while, do while, switch y for. Recuerde que estas subrutinas serán llamadas desde un programa principal.

```
!! while
! while (cond) { code; };
while:      ba cond
if_true:                                ! aca va el codigo
cond:       and cc %r1, %r1, %r0        ! r1 es la condicion
            bne if_true

!! do while
! do { code; } while (cond);
while:      ba code
code:                                   ! aca va el codigo
if_true:    andcc %r1, %r1, %r0         ! r1 es la condicion
            bne code

!! for
! for (set, cond, update) { code; };
for:        ba cond
cond:       andcc %r1, %r1, %r0
            bne code
code:                                   ! aca va el codigo
            ba update
update.                                 ' actualizo la variable
```

```
!! switch

! switch (var) { case n . (code) ; break ;
                 default . (code) ; break ; }

.macro      BEQUAL R1, R2, WHERE
            subcc R1, R2, %r0
            be WHERE
.end macro

switch:     ld var, %r1
switch-comp: BEQUAL %r1, cond-n, code-cond-n
            !...
            ba switch-default
code-cond-n . ! condition n code
            ! if break
            ba switch-end
            ! else
            ba switch-comp
            !...
switch-default : ! default case code
end.        sethi 0, %r0
```

3-Dado el siguiente código:

```
!-------------------------------------------------------------------------
!This declares an array of size 5 using literal definitions then
!stores it into an array of size 5 using .dwb

        .begin
        .org 2048
a_start .equ 3000
array_size .equ 5

2048) orcc %r0, array_size, %r1 !CLEARS THE FLAGS
2052) sub %r1, 1, %r1
2056) xor %r31, %i31, %g31
2060) sll %r1, 2, %r1
 loop:
2068) ld %r1, [a], %r5
2072) st %r5, %r1, [array]
2076) sub %r1, 4, %r1
2080) be done
2084)  subcc %r1, 0, %r0
2088) ba loop
 done:
        halt
3000) .org a_start
 array:  .dwb    array_size → .equ 5
 a:      47, -10, 33, -5, 7
3020)

        .end
!-------------------------------------------------------------------------
```

(a) Indicar la tabla de símbolos incluyendo relocalización y referencias externas,

(b) Señalar cuales son las líneas de este código que no tendrán representación en RAM cuando el programa sea cargado para su ejecución.

a)

symbol	value	global | external	relocatable
a_start	3000	no	no
array_size	5	no	no
a	3020	no	yes
array	3000	no	yes
done	2092	no	yes
loop	2064	no	yes

b) No tiene representación en RAM las directivas que son propias del ensamblador.