Contents

1)

a)

```
      SUBIS X0, X0, #0 // Compara X0 con 0
      B.LT else        // si es menor a cero va al else
      B done           // va al final del programa
else: SUB X0, XZR, X0  // cambia el signo de X0
done:
```

b)

```
        MOV X9, X0      // x9 = x0
        MOV X0, XZR     // x0 = 0
  loop: ADD X0, X0, X9  // x0 = x0 + x9
        SUBI X9, X9, #1 // x9 = x9 - 1
        CBNZ X9, loop   // va a loop si x9 != 0
  done:
```

Realiza la sumatoria $\sum_{i=1}^{X9} i$

2)

```
        SUBIS XZR, X9, #0  // Compara x9 con 0
        B.GE else          // si x9 >= 0 va al else
        B done             // termina la ejecucion
  else: ORRI X10, XZR, #2  // x10 = 0 | 2
  done:
```

a)

```
// {X9=0x0000000000101000}
        SUBIS XZR, X9, #0  // Compara x9 con 0
        B.GE else          // x9 >= 0 => va al else
        B done             //
  else: ORRI X10, XZR, #2  // x10 = 0 | 2 = 2
  done:
// {X10=0x0000000000000002}
```

b)

```
// {X9=0x8000000000001000}
        SUBIS XZR, X9, #0  // Compara x9 con 0
        B.GE else          // x9 < 0 => no hace nada
        B done             // Termina ejecucion
  else: ORRI X10, XZR, #2
  done:
// {X10=0x0000000000000001}
```

3) (C)

C

```
  if (i==N || j==N) {
    ++k;
  } else {
    ++i;
    ++j;
  }
```

LEGv8

```
        SUBS XZR, X0, X9 // compara i con N
        B.EQ true        // si es igual va a true
        SUBS XZR, X1, X9 // compara j con N
        B.EQ true        // si es igual va a true
        // Si no
        ADDI X0, X0, #1  // suma 1 a i
        ADDI X1, X1, #1  // suma 1 a j
        B DONE
  true: ADDI X2, X2, #1  // suma 1 a k
  done:
```

4)

Programa 1

LEGv8

```
loop: ADDI X0, X0, #2 // x0 = x0 + 2
      SUBI X1, X1, #1 // x1 = x1 - 1
      CBNZ X1, loop   // x1 != 0 => loop
done:
```

C

```
while (i != 0) {
  acc = acc + 2
  i = i - 1
}
```

Programa 2

LEGv8

```
loop: SUBIS X1, X1, #0 // compara x1 con 0
      B.LE done        // x1 <= 0 => done
      SUBI X1, X1, #1  // x1 = x1 - 1
      ADDI X0, X0, #2  // x0 = x0 + 2
      B loop           // va a loop
done:
```

C

```
while (i > 0) {
  i = i - 1
  acc = acc + 2
}
```

a)

Los valores finales de x0 son 2*10

b)

```
while (i != 0) {
  acc = acc + 2
  i = i - 1
}
```

c) (C)

Se ejecutan 3*N instrucciones LEGv8

d)

???

e)

```
while (i > 0) {
  i = i - 1
  acc = acc + 2
}
```

5)

1)

```
       ADD X10, XZR, XZR  // x10 = 0 + 0
loop: LDUR X1, [X0,#0]   // x1 = x0[0]
       ADD X2, X2, X1     // x2 = x2 + x1
       ADDI X0, X0, #8    // &x0 = &x0 + 1
       ADDI X10, X10, #1  // x10 = x10 + 1
       CMPI X10, #100     //
       B.LT loop          // x10 < 100 => loop
```

a)

El primer programa ejecuta $6 * 100 + 1$ instrucciones

b)

```
for (i = 0; i < 100; ++i) {
  a = memArray[i];
  result = result + a;
}
```

2)

```
       ADDI X10, XZR, #50  // x10 = 0 + 50
loop: LDUR X1, [X0,#0]    // x1 = x0[0]
       ADD X2, X2, X1      // x2 = x2 + x1
       LDUR X1, [X0,#8]    // x1 = x0[1]
       ADD X2, X2, X1      // x2 = x2 + x1
       ADDI X0, X0, #16    // x0 = &x0[2]
       SUBI X10, X10, #1   // x10 = x10 - 1
```

```
        CBNZ X10, loop      // x10 != 0 => loop
```

a)

El programa ejecuta $50 * 7 + 1$ veces

b)

```
for (i = 0; i < 50 ; i += 2) {
  a = memArray[i]
  result = result + a
  a = memArray[i+1]
  result = result + a
}
```

6)

Pseudocodigo

```
#define N (1<<10)
char *str;
long found, i;
for (found=0, i=0; i!=N; ++i)
found += (str[i]==48);

// Hace un LS y lo asigna a N
// Iniciar found=0
// Iniciar i=0
// Comparar i con N
// compara str[i] con 48
// str[i] == 48 => suma 1 a found
```