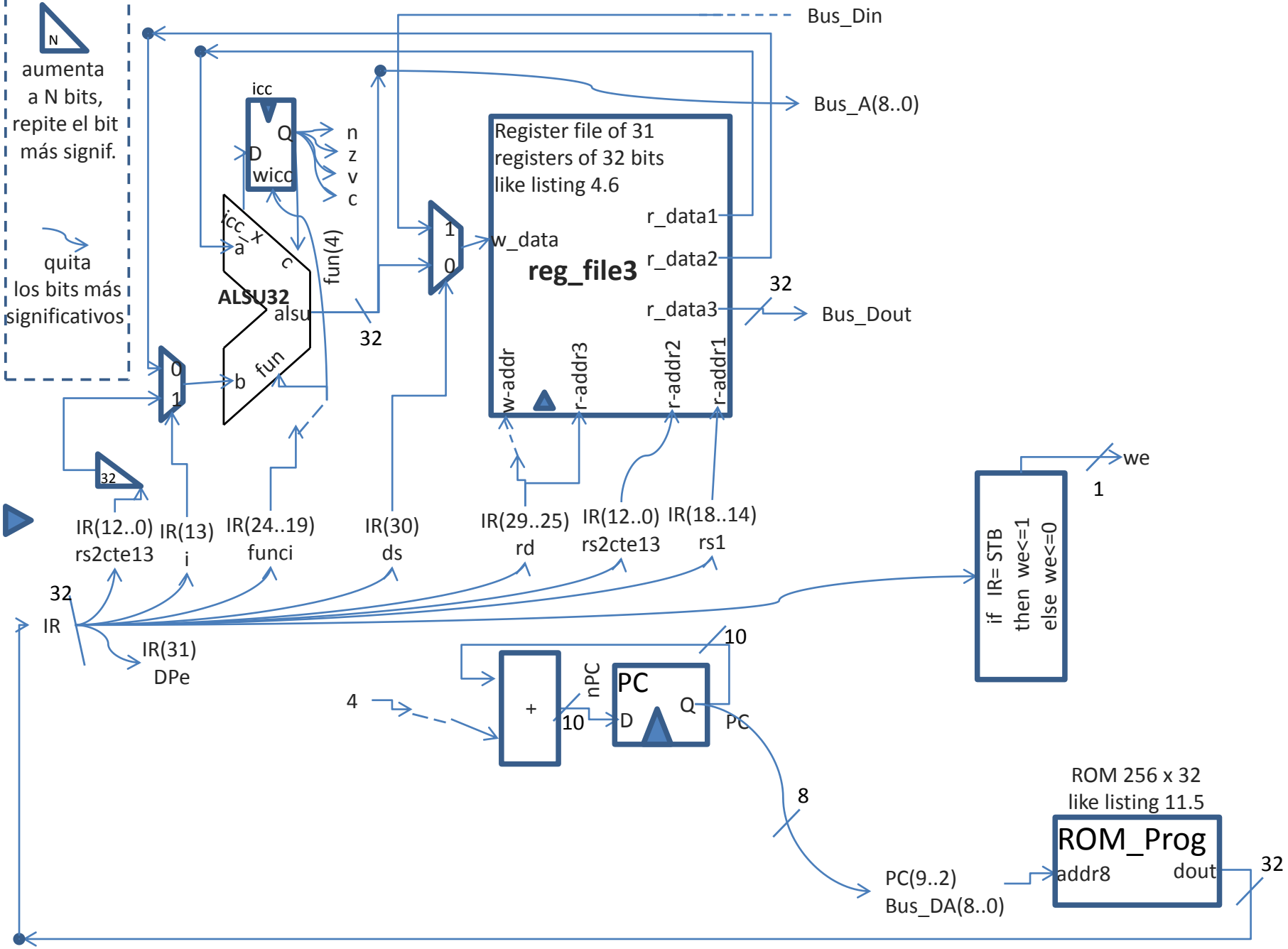
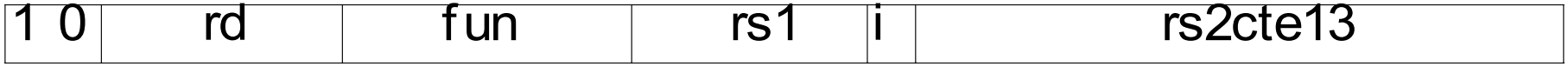
# parte del SPARCv8/4 y ROM



# Operaciones en la ALSU




| fun | FUN | Descripción |
|---|---|---|
| 0 | ADD | alu<=a+b |
| 1 | AND | alu<=a AND b |
| 2 | OR | alu<=a OR b |
| 3 | XOR | alu<=a XOR b |
| 4 | SUB | alu<=a - b |
| 5 | ANDN | alu<=a AND (NOT b) |
| 6 | ORN | alu<=a OR (NOT b) |
| 7 | XNOR | alu<=a XNOR b |
| 8 | ADDX | alu<=a + b + c |
| 9 | | |
| 10 | UMUL | alu<=a(15..0) * b(15..0) --(En el SPARC es de 32x32) |
| 11 | SMUL | alu<=a(15..0) * b(15..0) ) --(En el SPARC es de 32x32) |
| 12 | SUBX | alu<=a - b - c |
| 13 | | |
| 14 | UDIV | alu<=a / b(15..0) ) --(En el SPARC es de 64/32) |
| 15 | SDIV | alu<=a / b(15..0) --(En el SPARC es de 64/32) |
| | | |
| 37 | SLL | alu<=a<<b |
| 38 | SRL | alu<=a>>b |
| 39 | SRA | alu<=a/(2^b) |

| FUN rs1,rs2cte13,rd |
|---|
| r[rd]<- FUN(r[rs1],r[rs2cte13])<br>PC<-PC+4,<br>(Si es cc) icc<-icc_x |

# Primera instrucción ADD %g0,6,%g1

Cargar en %g1 el valor 6 (ADD %g0,6,%g1 )

- Cargar la constante 6 en rs2cte13,
- Se suma con cero en la ALU
- El resultado se almacena en el banco de registros, en el registro 1.
- Para ejecutarlo es necesario enviar un flaco positivo del reloj cuando las otras señales están en:
  - DPe<=’1’, que es para guardar en el banco de registros.
  - ds<='0'; porque no se va a usar la señal DPin.
  - rd<=”00001”; porque se va a almacenar en el registro 1.
  - fun<=”000000”; para realizar la suma.
  - rs1<=”00000”; para que sume 0 al 6 ya que r[0]=0.
  - i<='1'; para que cargue la constante 6.
  - rs2cte13<=”0000000000110”; que es el valor de la constate 6.

| DPe | ds | rd | fun | rs1 | i | rs2cte13 |
|---|---|---|---|---|---|---|
| 1 | 0 | 00001 | 000000 | 00000 | 1 | 0000000000110 |

# Direccionamiento del SPARCv8/4

- El SPARCv8/4 direcciona la memoria por bytes (conjunto de 8 bits).
- Sin embargo, las instrucciones son de 32 bits (es decir 4 bytes).
- Por lo tanto cada posición de memoria de programa tiene 4 bytes.
- Por este motivo el contador de programa debe contar de 4 en 4.

# Primer programa

- dir instrucción
- 0 MOV 6, %g1 ! Carga 6 en reg. global 1
- 4 MOV 9, %g2 ! Carga 9 en reg. global 2
- 8 ADD %g2, %g1, %g3 ! Suma en global 3
- 12 SRA %g3, 1, %g3 ! Desplaza en global 4

¿Qué operación aritmética hacen las dos últimas instrucciones?

```
1 00004000 82002006    MOV 6,%g1              ! Carga 6 en  reg. global 1
2 00004004 84002009    MOV 9,%g2              ! Carga 9 en  reg. global 2
3 00004008 86008001    ADD %g2, %g1, %g3      ! Suma y coloca en reg. global 3
4 0000400C 8938E001    SRA %g3,1, %g4         ! Desplaza y coloca en global 4
```

| SPARC V8 | Comentarios | Assembler | PC | R[rd] | | mem | icc | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 108 | Suma | ADD rs1,rs2cte13,rd | PC<-PC+4 | r[rd]<- r[rs1] + r[rs2]cte13 | | | | 1 | 0 | rd | | | | | 0 | 0 | 0 | 0 | 0 | 0 | rs1 | | | | i | rs2cte13 | | | | | | | | | | |
| 106 | Y | AND rs1,rs2cte13,rd | PC<-PC+4 | r[rd]<- r[rs1] AND r[rs2]cte13 | | | | 1 | 0 | rd | | | | | 0 | 0 | 0 | 0 | 0 | 1 | rs1 | | | | i | rs2cte13 | | | | | | | | | | |
| 106 | O | OR rs1,rs2cte13,rd | PC<-PC+4 | r[rd]<- r[rs1] OR r[rs2]cte13 | | | | 1 | 0 | rd | | | | | 0 | 0 | 0 | 0 | 1 | 0 | rs1 | | | | i | rs2cte13 | | | | | | | | | | |
| 106 | O excluyente | XOR rs1,rs2cte13,rd | PC<-PC+4 | r[rd]<- r[rs1] XOR r[rs2]cte13 | | | | 1 | 0 | rd | | | | | 0 | 0 | 0 | 0 | 1 | 1 | rs1 | | | | i | rs2cte13 | | | | | | | | | | |
| 110 | Resta | SUB rs1,rs2cte13,rd | PC<-PC+4 | r[rd]<- r[rs1] - r[rs2]cte13 | | | | 1 | 0 | rd | | | | | 0 | 0 | 0 | 1 | 0 | 0 | rs1 | | | | i | rs2cte13 | | | | | | | | | | |
| 106 | Y con rs2cte13 negada | ANDN rs1,rs2cte13,rd | PC<-PC+4 | r[rd]<- r[rs1] AND NOT(r[rs2]cte13) | | | | 1 | 0 | rd | | | | | 0 | 0 | 0 | 1 | 0 | 1 | rs1 | | | | i | rs2cte13 | | | | | | | | | | |
| 106 | O con rs2cte13 negada | ORN rs1,rs2cte13,rd | PC<-PC+4 | r[rd]<- r[rs1] OR NOT(r[rs2])cte13 | | | | 1 | 0 | rd | | | | | 0 | 0 | 0 | 1 | 1 | 0 | rs1 | | | | i | rs2cte13 | | | | | | | | | | |
| 106 | O excluyente negada | XNOR rs1,rs2cte13,rd | PC<-PC+4 | r[rd]<- r[rs1] XNOR r[rs2]cte13 | | | | 1 | 0 | rd | | | | | 0 | 0 | 0 | 1 | 1 | 1 | rs1 | | | | i | rs2cte13 | | | | | | | | | | |
| 108 | Suma con acarreo | ADDX rs1,rs2cte13,rd | PC<-PC+4 | r[rd]<- r[rs1] + r[rs2]cte13 + c | | | | 1 | 0 | rd | | | | | 0 | 0 | 1 | 0 | 0 | 0 | rs1 | | | | i | rs2cte13 | | | | | | | | | | |
| 113 | Mult. sin signo | UMUL rs1,rs2cte13,rd | PC<-PC+4 | r[rd]<- r[rs1] * r[rs2]cte13 | | | | 1 | 0 | rd | | | | | 0 | 0 | 1 | 0 | 1 | 0 | rs1 | | | | i | rs2cte13 | | | | | | | | | | |
| 113 | Mult. con signo | SMUL rs1,rs2cte13,rd | PC<-PC+4 | r[rd]<- r[rs1] * r[rs2]cte13 | | | | 1 | 0 | rd | | | | | 0 | 0 | 1 | 0 | 1 | 1 | rs1 | | | | i | rs2cte13 | | | | | | | | | | |
| 110 | Resta con acarreo | SUBX rs1,rs2cte13,rd | PC<-PC+4 | r[rd]<- r[rs1] - r[rs2]cte13 - c | | | | 1 | 0 | rd | | | | | 0 | 0 | 1 | 1 | 0 | 0 | rs1 | | | | i | rs2cte13 | | | | | | | | | | |
| 115 | Div. sin signo | UDIV rs1,rs2cte13,rd | PC<-PC+4 | r[rd]<- r[rs1] / r[rs2]cte13 | | | | 1 | 0 | rd | | | | | 0 | 0 | 1 | 1 | 1 | 0 | rs1 | | | | i | rs2cte13 | | | | | | | | | | |
| 115 | Div. con signo | SDIV rs1,rs2cte13,rd | PC<-PC+4 | r[rd]<- r[rs1] / r[rs2]cte13 | | | | 1 | 0 | rd | | | | | 0 | 0 | 1 | 1 | 1 | 1 | rs1 | | | | i | rs2cte13 | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 108 | Suma | ADDcc rs1,rs2cte13,rd | PC<-PC+4 | r[rd]<- r[rs1] + r[rs2]cte13 | | | icc<-icc_x | 1 | 0 | rd | | | | | 0 | 1 | 0 | 0 | 0 | 0 | rs1 | | | | i | rs2cte13 | | | | | | | | | | |
| 106 | Y | ANDcc rs1,rs2cte13,rd | PC<-PC+4 | r[rd]<- r[rs1] AND r[rs2]cte13 | | | icc<-icc_x | 1 | 0 | rd | | | | | 0 | 1 | 0 | 0 | 0 | 1 | rs1 | | | | i | rs2cte13 | | | | | | | | | | |
| 106 | O | ORcc rs1,rs2cte13,rd | PC<-PC+4 | r[rd]<- r[rs1] OR r[rs2]cte13 | | | icc<-icc_x | 1 | 0 | rd | | | | | 0 | 1 | 0 | 0 | 1 | 0 | rs1 | | | | i | rs2cte13 | | | | | | | | | | |
| 106 | O excluyente | XORcc rs1,rs2cte13,rd | PC<-PC+4 | r[rd]<- r[rs1] XOR r[rs2]cte13 | | | icc<-icc_x | 1 | 0 | rd | | | | | 0 | 1 | 0 | 0 | 1 | 1 | rs1 | | | | i | rs2cte13 | | | | | | | | | | |
| 110 | Resta | SUBcc rs1,rs2cte13,rd | PC<-PC+4 | r[rd]<- r[rs1] - r[rs2]cte13 | | | icc<-icc_x | 1 | 0 | rd | | | | | 0 | 1 | 0 | 1 | 0 | 0 | rs1 | | | | i | rs2cte13 | | | | | | | | | | |
| 106 | Y con rs2cte13 negada | ANDNcc rs1,rs2cte13,rd | PC<-PC+4 | r[rd]<- r[rs1] AND NOT(r[rs2]cte13) | | | icc<-icc_x | 1 | 0 | rd | | | | | 0 | 1 | 0 | 1 | 0 | 1 | rs1 | | | | i | rs2cte13 | | | | | | | | | | |
| 106 | O con rs2cte13 negada | ORNcc rs1,rs2cte13,rd | PC<-PC+4 | r[rd]<- r[rs1] OR NOT(r[rs2])cte13 | | | icc<-icc_x | 1 | 0 | rd | | | | | 0 | 1 | 0 | 1 | 1 | 0 | rs1 | | | | i | rs2cte13 | | | | | | | | | | |
| 106 | O excluyente negada | XNORcc rs1,rs2cte13,rd | PC<-PC+4 | r[rd]<- r[rs1] XNOR r[rs2]cte13 | | | icc<-icc_x | 1 | 0 | rd | | | | | 0 | 1 | 0 | 1 | 1 | 1 | rs1 | | | | i | rs2cte13 | | | | | | | | | | |
| 108 | Suma con acarreo | ADDXcc rs1,rs2cte13,rd | PC<-PC+4 | r[rd]<- r[rs1] + r[rs2]cte13 + c | | | icc<-icc_x | 1 | 0 | rd | | | | | 0 | 1 | 1 | 0 | 0 | 0 | rs1 | | | | i | rs2cte13 | | | | | | | | | | |
| 113 | Mult. sin signo | UMULcc rs1,rs2cte13,rd | PC<-PC+4 | r[rd]<- r[rs1] * r[rs2]cte13 | | | icc<-icc_x | 1 | 0 | rd | | | | | 0 | 1 | 1 | 0 | 1 | 0 | rs1 | | | | i | rs2cte13 | | | | | | | | | | |
| 113 | Mult. con signo | SMULcc rs1,rs2cte13,rd | PC<-PC+4 | r[rd]<- r[rs1] * r[rs2]cte13 | | | icc<-icc_x | 1 | 0 | rd | | | | | 0 | 1 | 1 | 0 | 1 | 1 | rs1 | | | | i | rs2cte13 | | | | | | | | | | |
| 110 | Resta con acarreo | SUBXcc rs1,rs2cte13,rd | PC<-PC+4 | r[rd]<- r[rs1] - r[rs2]cte13 - c | | | icc<-icc_x | 1 | 0 | rd | | | | | 0 | 1 | 1 | 1 | 0 | 0 | rs1 | | | | i | rs2cte13 | | | | | | | | | | |
| 115 | Div. sin signo | UDIVcc rs1,rs2cte13,rd | PC<-PC+4 | r[rd]<- r[rs1] / r[rs2]cte13 | | | icc<-icc_x | 1 | 0 | rd | | | | | 0 | 1 | 1 | 1 | 1 | 0 | rs1 | | | | i | rs2cte13 | | | | | | | | | | |
| 115 | Div. con signo | SDIVcc rs1,rs2cte13,rd | PC<-PC+4 | r[rd]<- r[rs1] / r[rs2]cte13 | | | icc<-icc_x | 1 | 0 | rd | | | | | 0 | 1 | 1 | 1 | 1 | 1 | rs1 | | | | i | rs2cte13 | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 107 | Desp. a la izq. (llena con ceros) | SLL rs1,rs2cte13,rd | PC<-PC+4 | r[rd]<- r[rs1] << r[rs2]cte13 | | | | 1 | 0 | rd | | | | | 1 | 0 | 0 | 1 | 0 | 1 | rs1 | | | | i | rs2cte13 | | | | | | | | | | |
| 107 | Desp. a la der. (llena con ceros) | SRL rs1,rs2cte13,rd | PC<-PC+4 | r[rd]<- r[rs1] >> r[rs2]cte13 | | | | 1 | 0 | rd | | | | | 1 | 0 | 0 | 1 | 1 | 0 | rs1 | | | | i | rs2cte13 | | | | | | | | | | |
| 107 | Desp. a la der. (llena con signo) | SRA rs1,rs2cte13,rd | PC<-PC+4 | r[rd]<- r[rs1] / (2**r[rs2]cte13) | | | | 1 | 0 | rd | | | | | 1 | 0 | 0 | 1 | 1 | 1 | rs1 | | | | i | rs2cte13 | | | | | | | | | | |

# INSTRUCCIONES 1/2
# SPARC v8/4

| SPARC V8 | Comentarios | Assembler | PC | R[rd] | mem | icc | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 90 | Carga en reg. dato de MEM | LD [rs1+rs2],rd | PC<-PC+4 | r[rd]<-m[r[rs1]+r[rs2]] | | | 1 | 1 | rd | | | | | 0 | 0 | 0 | 0 | 0 | 0 | rs1 | | | | | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | rs2 | | | | |
| 90 | Carga en reg. dato de MEM | LD [rs1+cte13],rd | PC<-PC+4 | r[rd]<-m[r[rs1]+cte13] | | | 1 | 1 | rd | | | | | 0 | 0 | 0 | 0 | 0 | 0 | rs1 | | | | | 1 | cte13 | | | | | | | | | | | | |
| 95 | Almacena el reg. en la MEM | ST rd,[rs1+rs2] | PC<-PC+4 | | m[r[rs1]+r[rs2]]<-r[rd] | | 1 | 1 | rd | | | | | 0 | 0 | 0 | 1 | 0 | 0 | rs1 | | | | | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | rs2 | | | | |
| 95 | Almacena el reg. en la MEM | ST rd,[rs1+cte13] | PC<-PC+4 | | m[r[rs1]+cte13]<-r[rd] | | 1 | 1 | rd | | | | | 0 | 0 | 0 | 1 | 0 | 0 | rs1 | | | | | 1 | cte13 | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 119 | No brinca | BN cte22 | PC<-PC+4*cte22 WHEN '0'='1' ELSE PC+4 | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | | | | | | | | | | | | | | | | | | | | | | | | |
| 119 | Brinca si son iguales | BE cte22 | PC<-PC+4*cte22 WHEN z='1' ELSE PC+4 | | | | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | | | | | | | | | | | | | | | | | | | | | | | | |
| 119 | Brinca si es menor o igual (S) | BLE cte22 | PC<-PC+4*cte22 WHEN zOR(nXORv)='1' ELSE PC+4 | | | | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | | | | | | | | | | | | | | | | | | | | | | | | |
| 119 | Brinca si es menor (S) | BL cte22 | PC<-PC+4*cte22 WHEN nXORv='1' ELSE PC+4 | | | | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | | | | | | | | | | | | | | | | | | | | | | | | |
| 119 | Brinca si es menor o igual (U) | BLEU cte22 | PC<-PC+4*cte22 WHEN cORz='1' ELSE PC+4 | | | | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | | | | | | | | | | | | | | | | | | | | | | | | |
| 119 | Brinca si c es uno (menor (U)) | BCS cte22 | PC<-PC+4*cte22 WHEN c='1' ELSE PC+4 | | | | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | | | | | | | | | | | | | | | | | | | | | | | | |
| 119 | Brinca si es negativo | BNEG cte22 | PC<-PC+4*cte22 WHEN n='1' ELSE PC+4 | | | | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | | | | | | | | | | | | | | | | | | | | | | | | |
| 119 | Brinca si v es uno | BVS cte22 | PC<-PC+4*cte22 WHEN v='1' ELSE PC+4 | | | | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | | | | | | | | | | | | | | | | | | | | | | | | |
| 119 | Brinca siempre | BA cte22 | PC<-PC+4*cte22 WHEN '1'='1' ELSE PC+4 | | | | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | | | | | | | | | | | | | | | | | | | | | | | | |
| 119 | Brinca si no son iguales | BNE cte22 | PC<-PC+4*cte22 WHEN NOTz='1' ELSE PC+4 | | | | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | | | | | | | | | | | | | | | | | | | | | | | | |
| 119 | Brinca si es mayor (S) | BG cte22 | PC<-PC+4*cte22 WHEN NOT(zOR(nXORv))='1' ELSE PC+4 | | | | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | | | | | | | | | | | | | | | | | | | | | | | | |
| 119 | Brinca si es mayor o igual (S) | BGE cte22 | PC<-PC+4*cte22 WHEN NOT(nXORv)='1' ELSE PC+4 | | | | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | | | | | | | | | | | | | | | | | | | | | | | | |
| 119 | Brinca si es mayor (U) | BGU cte22 | PC<-PC+4*cte22 WHEN NOT(cORz)='1' ELSE PC+4 | | | | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | | | | | | | | | | | | | | | | | | | | | | | | |
| 119 | Brinca si c es cero (may,Igu (U)) | BCC cte22 | PC<-PC+4*cte22 WHEN NOTc='1' ELSE PC+4 | | | | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | | | | | | | | | | | | | | | | | | | | | | | | |
| 119 | Brinca si es positivo | BPOS cte22 | PC<-PC+4*cte22 WHEN NOTn='1' ELSE PC+4 | | | | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | | | | | | | | | | | | | | | | | | | | | | | | |
| 119 | Brinca si v es cero | BVC cte22 | PC<-PC+4*cte22 WHEN NOTv='1' ELSE PC+4 | | | | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | | | | | | | | | | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 125 | Llamado a rutina | CALL cte30 | PC<-PC+4*cte30 %o7<-PC | | | | 0 | 1 | cte30 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 85 | Retorno de rutina | RETL | PC<-%o7+8 | | | | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 126 | Salto con reg. | JMPL rs1+rs2,rd | PC<-r[rs1]+r[rs2] r[rd]<-PC | | | | 1 | 0 | rd | | | | | 1 | 1 | 1 | 0 | 0 | 0 | rs1 | | | | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | rs2 | | | | |
| 126 | Salto con cte. | JMPL rs1+cte13,rd | PC<-r[rs1]+cte13 r[rd]<-PC | | | | 1 | 0 | rd | | | | | 1 | 1 | 1 | 0 | 0 | 0 | rs1 | | | | | 1 | cte13 | | | | | | | | | | | | |

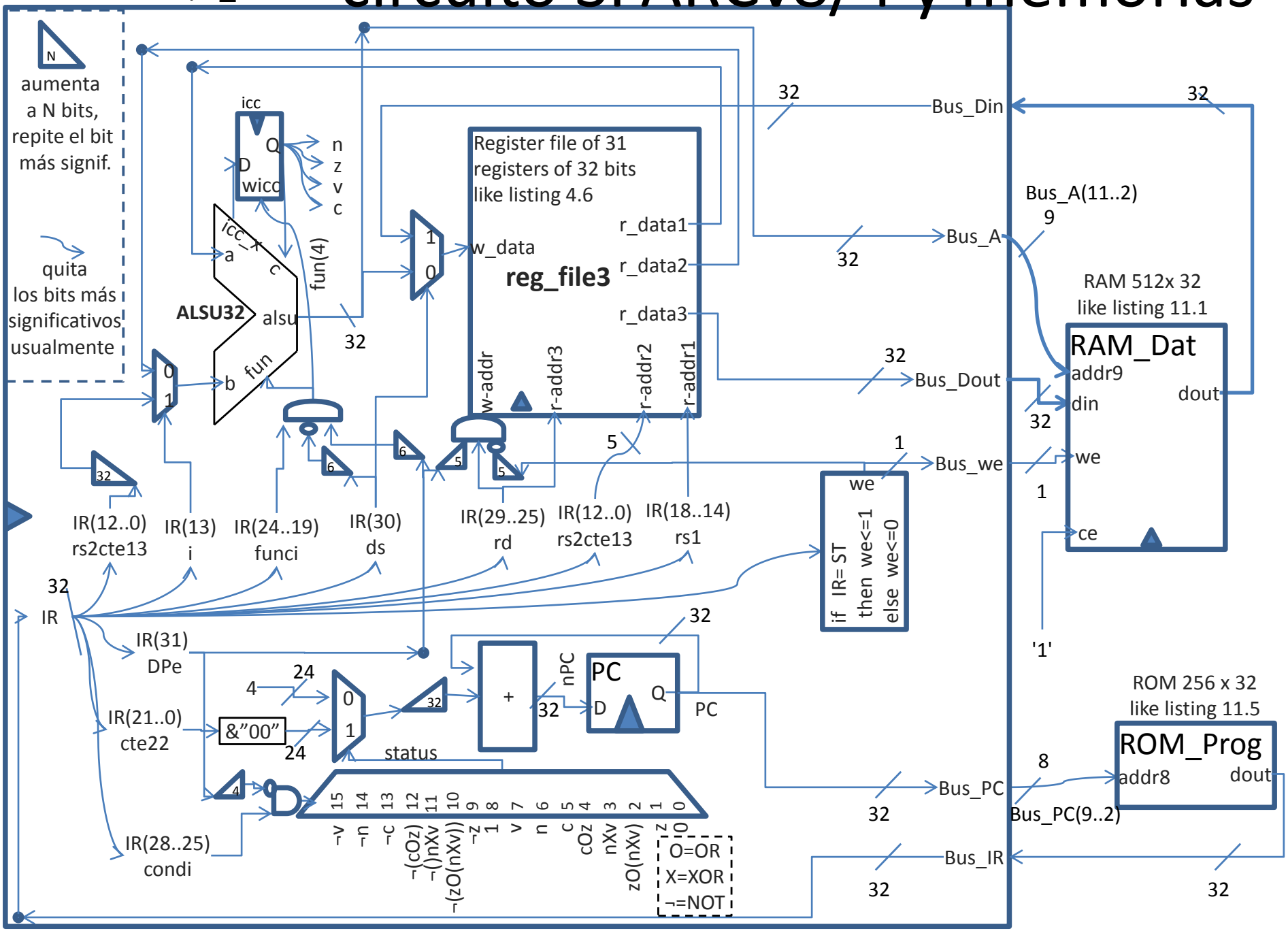
# INSTRUCCIONES 2/2
# SPARC v8/4

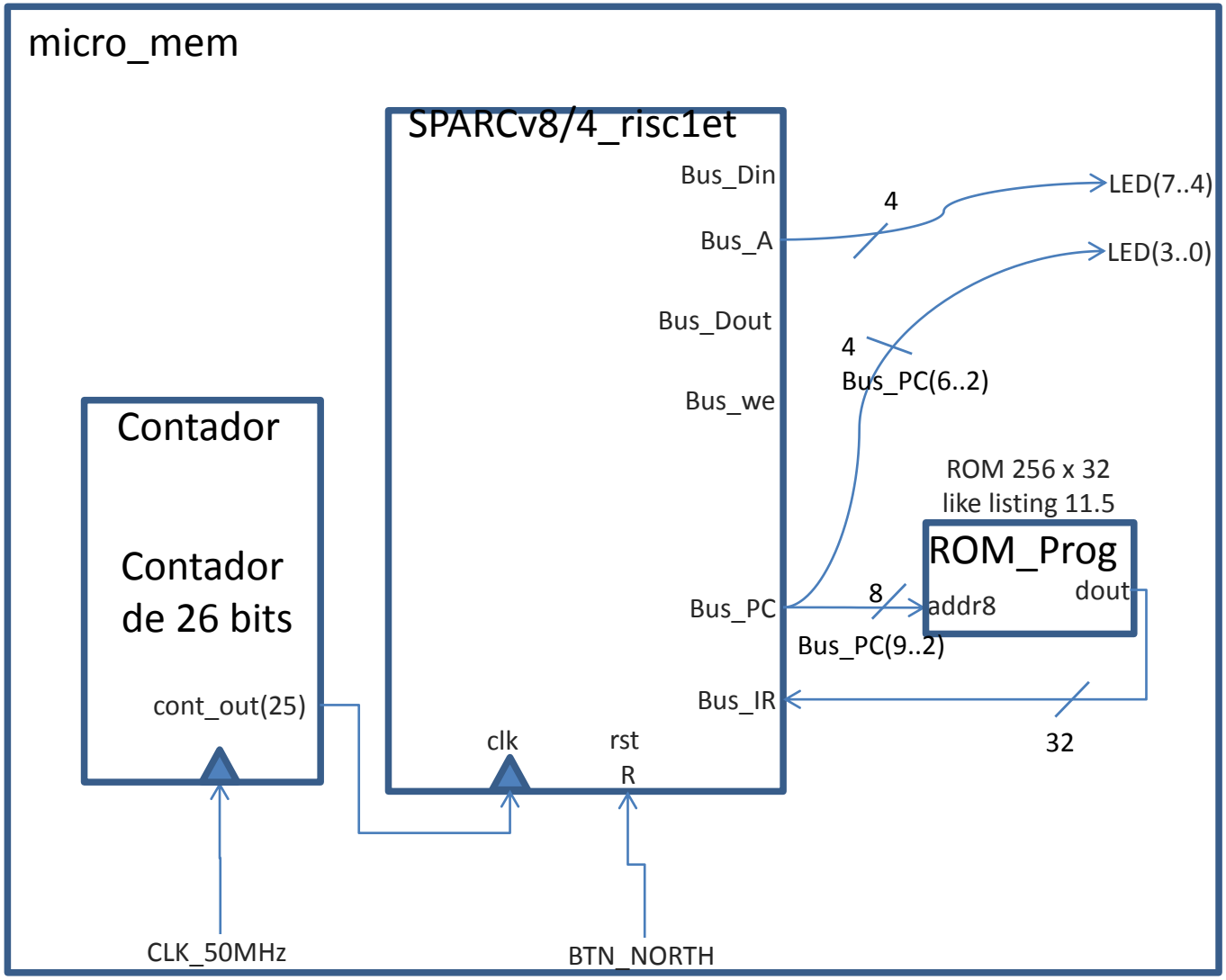
| | | |
|---|---|---|
| No hace operaciones | NOP | ADD R0,R0,R0 |
| Mueve | MOV rs2cte13,rd | ADD R0,rs2cte13,rd |
| Compara mayo o menor | CMPcc rs1,rs2cte13 | SUBcc rs1,rs2cte13,R0 |
| Compara con cero | TSTcc rs2 | ORcc R0,rs2,R0 |
| Invertir los bits | NOT rs1,rd | XNOR rs1,R0,rd |
| Complemento a 2 | NEG rs2,rd | SUBB R0,rs2,rd |
| Incrementa en uno | INC rs1,rd | ADD rs1,1,rd |
| Decrementa en 1 | DEC rs1,rd | SUBB rs1,1,rd |

# SIMULACIÓN SPARC v8/4

| Ciclos | PC | n,z,v,c | %l1 | %l2 | %l3 | %l4 | | | | | ir | DPe | ds | | rs1,r-addr1 | r-data1,a | rs2cte13,r-addr2 | r-data2 | i | b | funci | fun | alsu, Bus_A | Bus_Din | rd,r-addr3 | r-data3, Bus_Dout | w-addr | w-data | we | | condi | status | cte22 | nPC |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 3 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 4 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 5 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 6 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 7 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 8 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 9 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 10 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 11 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 12 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 13 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 14 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 15 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 16 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 17 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 18 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 19 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 20 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 21 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 22 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 23 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 24 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 25 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 26 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 27 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 28 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 29 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

# SPARCv8/4_risc1et circuito SPARCv8/4 y memorias




micro_mem
SPARCv8/4_risc1et
Bus_Din
Bus_A
Bus_Dout
Bus_we
Bus_PC
Bus_IR
clk
rst
R
LED(7..4)
LED(3..0)
4
4
Bus_PC(6..2)
ROM 256 x 32
like listing 11.5
ROM_Prog
addr8
dout
8
Bus_PC(9..2)
32
Contador
Contador
de 26 bits
cont_out(25)
CLK_50MHz
BTN_NORTH

# Lectura de memoria

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 1 | rd | 0 0 0 0 0 0 | rs1 | 0 | 0 0 0 0 0 1 0 1 0 | rs2 |
| 1 1 | rd | 0 0 0 0 0 0 | rs1 | 1 | cte13 | |

| LD [rs1+rs2cte13],rd |
|---|
| r[rd]<- m[r[rs1]+r[rs2cte13]]<br>PC<-PC+4 |

# Escritura de memoria

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 1 | rd | 0 0 0 1 0 0 | rs1 | 0 | 0 0 0 0 0 1 0 1 0 | rs2 |
| 1 1 | rd | 0 0 0 1 0 0 | rs1 | 1 | cte13 | |

| ST rd,[rs1+rs2cte13] |
|---|
| m[r[rs1]+r[rs2cte13]]<- r[rd]<br>PC<-PC+4 |

```
4 00004000 E0002064    LD [%g0+100],%l0
5 00004004 A0042005    ADD %l0,5,%l0
6 00004008 E0202068    st %l0,[%g0+104]
```

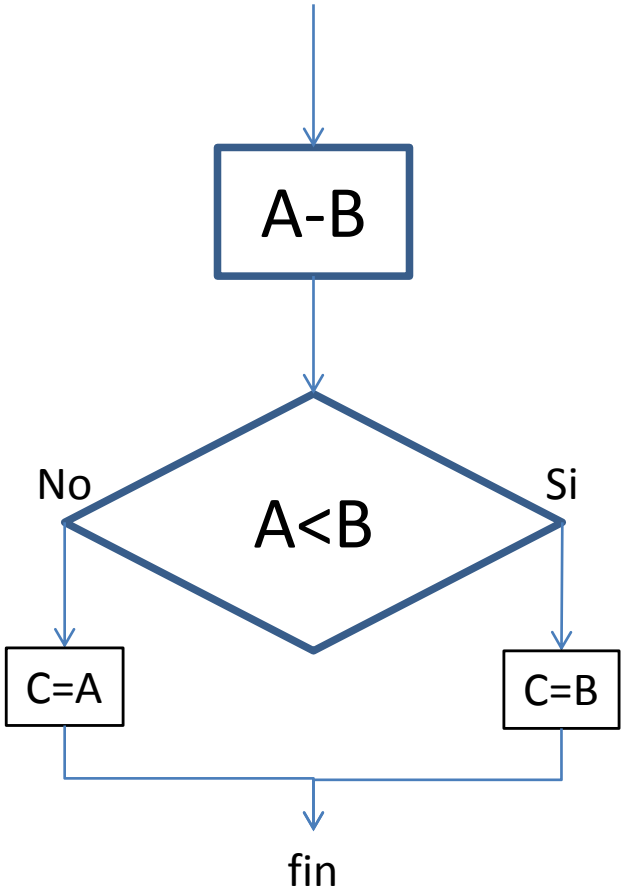
# Ejercicio

$$m[500] = \sum_{n=0}^{3} m[4 * n + 100]$$

$$m[500] = \sum_{n=0}^{99} m[4 * n + 100]$$

# Máximo

%o0 : A
%o1 : B
%o3 : C




```
max:          SUBcc %o0,%o1,%g0
              BL Si_cumple
              NOP

No_cumple:    MOV %o0,%o3
              BA fin
              NOP

Si_cumple:    MOV %o1,%o3

fin:          NOP
```

# Brincos

Bcondi cte22

PC<-PC+4*cte22 WHEN condi ELSE PC+4

| 0 0 0 | condi | 0 1 0 | cte22 |
|---|---|---|---|

Código de la condición

Ecuación de la condición

Nombre lógico de la condición {N,Z,V,C}x{Set, Clear}

Interpretación aritmética con A-B

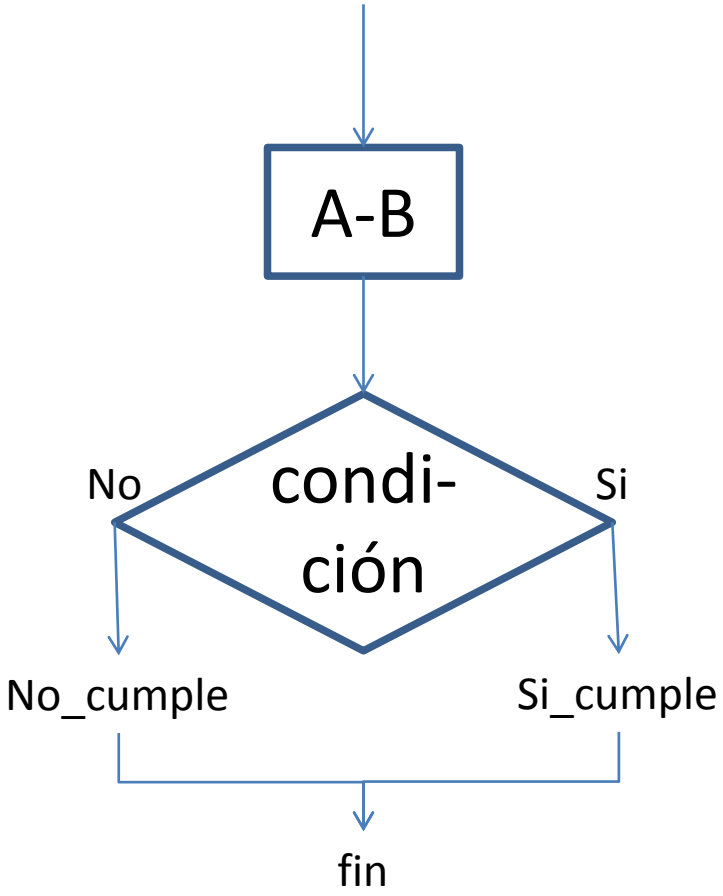
Nombre aritmético de la condición

Nombre del SPARC de la condición

En el SPARC V8 antes de saltar se ejecuta la instrucción siguiente, por eso llevan NOP. En el SPARC V8/4 no.

| condi | status(icc,condi) | CONDI(L) | A-B | CONDI(A) | CONDI | Descrip. (Inglés) |
|---|---|---|---|---|---|---|
| 0000 | 0 | BN | 1=0 ! | | BN | Branch Never |
| 0001 | Z | BZS | A=B | BE | BE | Branch on Equal |
| 0010 | Z or (N xor V) | | A≤B (S) | BLE | BLE | Branch on Less or Equal |
| 0011 | N xor V | | A<B (S) | BL | BL | Branch on Less |
| 0100 | (CorZ) | | A≤B (U) | BLEU | BLEU | Branch on Less or Equal Unsigned |
| 0101 | C | BCS | A<B (U) | BLU | BCS | Branch on Carry Set (Less than, Unsigned) |
| 0110 | N | BNS | | BNEG | BNEG | Branch on Negative |
| 0111 | V | BVS | | | BVS | Branch on Overflow Set |
| 1000 | 1 | BA | 0=0 | | BA | Branch Always |
| 1001 | not Z | BZC | A≠B | BNE | BNE | Branch on Not Equal |
| 1010 | not (Z or (N xor V)) | | A>B (S) | BG | BG | Branch on Greater |
| 1011 | not (N xor V) | | A≥B (S) | BGE | BGE | Branch on Greater or Equal |
| 1100 | not (C or Z) | | A>B (U) | BGU | BGU | Branch on Greater Unsigned |
| 1101 | not C | BCC | A≥B (U) | BGEU | BCC | Branch on Carry Clear (Greater than or Equal, U.) |
| 1110 | not N | BNC | | BPOS | BPOS | Branch on Positive |
| 1111 | not V | BVC | | | BVC | Branch on Overflow Clear |

# Estructura IF




```
                    SUBcc A,B,%g0
                    Bcondi Si_cumple
No_cumple:          ...
                    ...
                    BA fin
Si_cumple:          ...
                    ...
fin:                ...
```

# %g3 = %g1 ** %g2

```
--R1^2
MOV 1, %g3
SMUL %g3, %g1, %g3
SMUL %g3, %g1, %g3
```

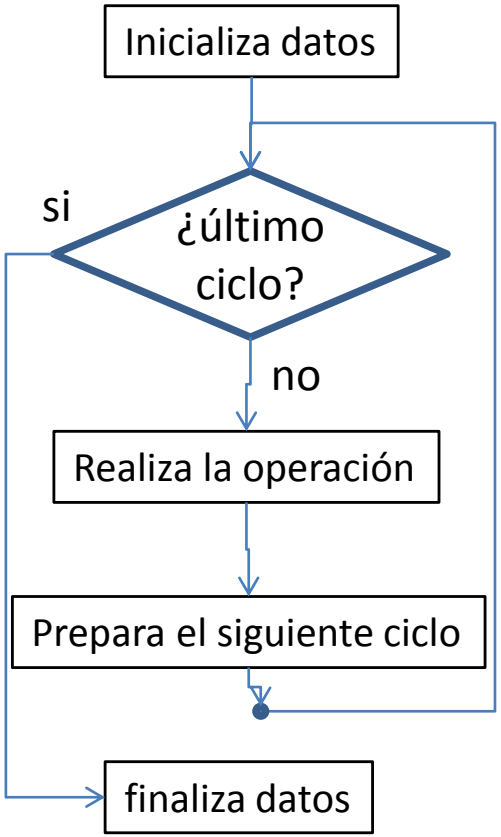
```
--R1^3
MOV 1, %g3
SMUL %g3, %g1, %g3
SMUL %g3, %g1, %g3
SMUL %g3, %g1, %g3
```

```
      --R1^n
    MOV 1, %g3
      --repetir n veces
0  SMUL %g3, %g1, %g3
1  SMUL %g3, %g1, %g3
      .
      .
      .
n-1  SMUL %g3, %g1, %g3
n     --termina
```

```
        --R1^R2
        --inicializa
exp     MOV 1, %g3 – lleva el producto
        MOV 0, %g4– cuenta productos
        --termino?
ciclo   CMPcc %g4, %g2
        BE fin
        NOP
        --realiza la operación
        SMUL %g3, %g1, %g3
        --prepara el siguiente ciclo
        INC %g4, %g4
        -- repite
        BA ciclo
        NOP
fin
```

# Brincos y ciclo WHILE (1/3)

$$y = e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2} + \frac{x \cdot x^2}{3 \cdot 2} + \cdots + \frac{x^{i-1}}{(i-1)!} + \frac{x \cdot x^{i-1}}{i \cdot (i-1)!} + \cdots$$

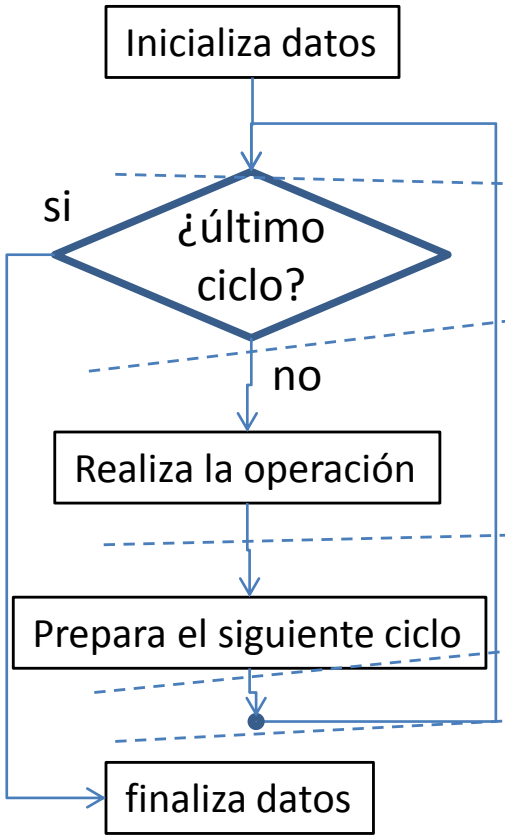



R1=x tiene la entrada que se lee de la memoria 100
R2=n es un contador, comienza con 1.
R3=p=x^n, se calcula mult. x al valor anterior de p
R4=f=n!, se calcula mult. n al valor anterior de f
R5=d=p/f
R6=y=e^x se calcula sumando d al valor anterior de y y se debe almacenar en la memoria 101.

El ciclo se detiene cuando d=0

# Brincos y ciclo WHILE (2/3)

$$y = e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2} + \frac{x \cdot x^2}{3 \cdot 2} + \cdots + \frac{x^{i-1}}{(i-1)!} + \frac{x \cdot x^{i-1}}{i \cdot (i-1)!} + \cdots$$




```
1  x=evstr(x_dialog('Leyendo x de MEM(100)','0')) //R1
2  //Pregunta el valor de x en un cuadro de diálogo.
3  n=1 //R2 contador
4  p=1 //R3 numerador con potencias de x
5  f=1 //R4 denominador con factorial de n
6  d=1 //R5 división
7  y=1 //R6 sumas parciales
8
9  while (d~=0),
10
11 p=p*x;
12 f=f*n;
13 d=int(p/f);
14 y=y+d;
15
16 n=n+1
17
18 end;
19
20 x_dialog(["Escribiendo y en MEM[101]",string(y)])
21 //Muestra el valor de y en un cuadro de diálogo
```

# Brincos y ciclo WHILE (3/3)

$$y = e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2} + \frac{x \cdot x^2}{3 \cdot 2} + \cdots + \frac{x^{i-1}}{(i-1)!} + \frac{x \cdot x^{i-1}}{i \cdot (i-1)!} + \cdots$$

```
1                       .global _start
2                       _start:
3  00004000 92002005            MOV 5, %o1
4
5                       exp:    ! %o1 = n
6                               ! %o2 = e^n
7  00004004 A4002001            MOV 1, %l2
8  00004008 A6002001            MOV 1, %l3
9  0000400C A8002001            MOV 1, %l4
10 00004010 AA002001            MOV 1, %l5
11 00004014 94002001            MOV 1, %o2
12
13 00004018 80954000    ciclo:  TST %l5
14 0000401C 02800009            BE fin
15 00004020 80000000            NOP
16
17 00004024 A65CC009          SMUL %l3, %o1, %l3
18 00004028 A85D0012          SMUL %l4, %l2, %l4
19 0000402C AA7CC014          SDIV %l3, %l4, %l5
20 00004030 94028015          ADD  %o2, %l5, %o2
21
22 00004034 A404A001          ADD %l2,1, %l2
23
24 00004038 10BFFFF8            BA ciclo
25 0000403C 80000000            NOP
26
27 00004040 80000000    fin:    NOP
```

```
1  x=evstr(x_dialog('Leyendo x de MEM(100)','0')) //R
2  //Pregunta el valor de x en un cuadro de diálogo.
3  n=1 //R2 contador
4  p=1 //R3 numerador con potencias de x
5  f=1 //R4 denominador con factorial de n
6  d=1 //R5 división
7  y=1 //R6 sumas parciales
8
9  while (d~=0),
10
11 p=p*x;
12 f=f*n;
13 d=int(p/f);
14 y=y+d;
15
16 n=n+1
17
18 end;
19
20 x_dialog(["Escribiendo y en MEM[101]",string(y)])
21 //Muestra el valor de y en un cuadro de diálogo
```

# Brincos y ciclo WHILE (3/3)

$$y = e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2} + \frac{x \cdot x^2}{3 \cdot 2} + \cdots + \frac{x^{i-1}}{(i-1)!} + \frac{x \cdot x^{i-1}}{i \cdot (i-1)!} + \cdots$$

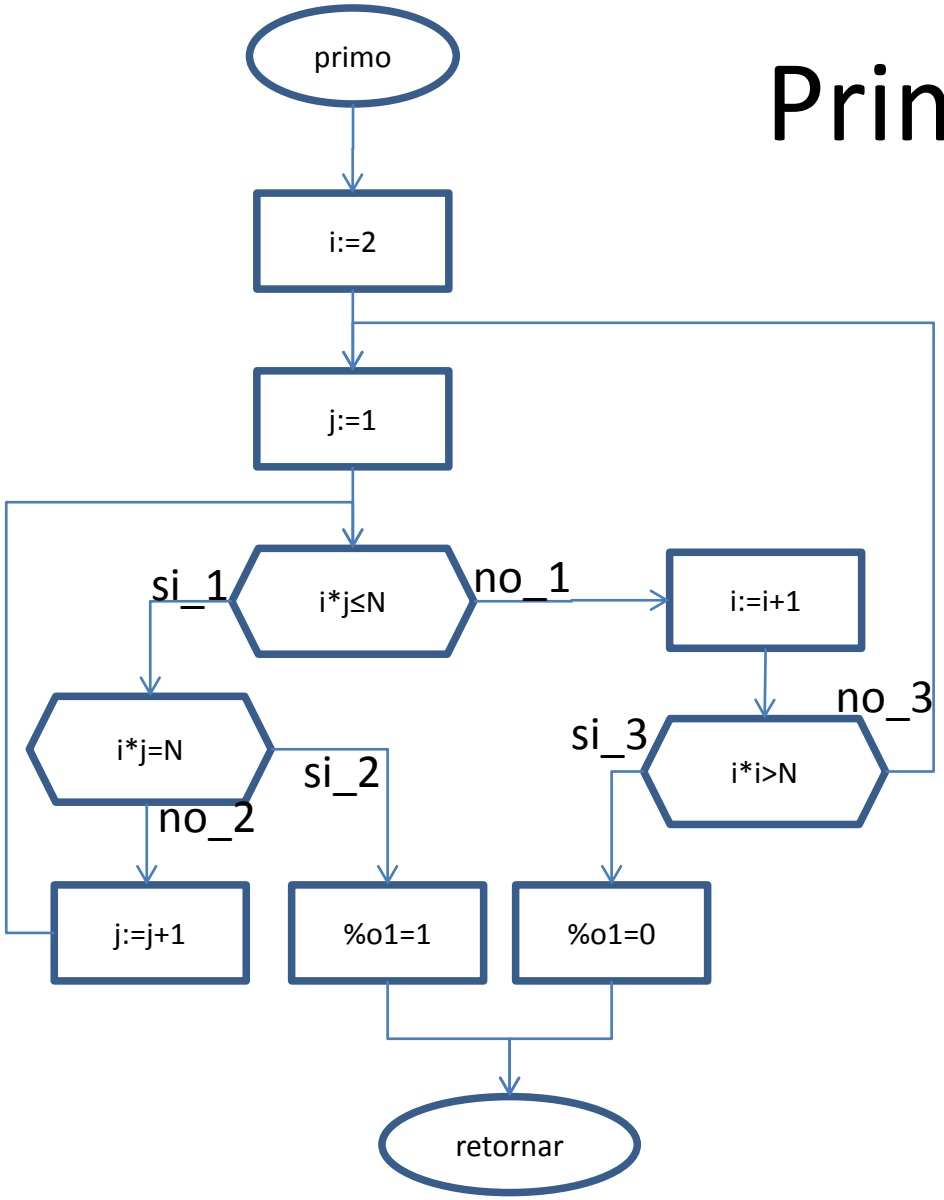
```
0 inicio:  LD [%g0+100],%g1
4          MOV 1, %l2
8          MOV 1, %l3
12         MOV 1, %l4
16         MOV 1, %l5
20         MOV 1, %l6

24 ciclo:  TST %g5
28         Bz fin (9)
32         NOP

36         SMUL %g3, %g1, %g3
40         SMUL %g4, %g2, %g4
44         SDIV %g3, %g4, %g5
48         ADD %g6, %g5, %g6

52         INC %g2, %g2

56         BA ciclo (-8)
60         NOP

64 fin:    ST  %g6,[%g0+101]
```

```
1  x=evstr(x_dialog('Leyendo x de MEM(100)','0')) //R1
2  //Pregunta el valor de x en un cuadro de diálogo.
3  n=1 //R2 contador
4  p=1 //R3 numerador con potencias de x
5  f=1 //R4 denominador con factorial de n
6  d=1 //R5 división
7  y=1 //R6 sumas parciales
8
9  while (d~=0),
10
11 p=p*x;
12 f=f*n;
13 d=int(p/f);
14 y=y+d;
15
16 n=n+1
17
18 end;
19
20 x_dialog(["Escribiendo y en MEM[101]",string(y)])
21 //Muestra el valor de y en un cuadro de diálogo
```

# Ejercicio de brincos

- Determinar si el valor de %o0 es un número primo.
  - Recuerde que los números primos son múltiplos únicamente de él y de 1.
  - Además 1 no se considera primo.
  - Puede ir probando cada número i hasta que i^2 sea mayor que %o0.
- En %o1 debe salir un 0 si %o0 es primo, de lo contrario un 1.

# Primo



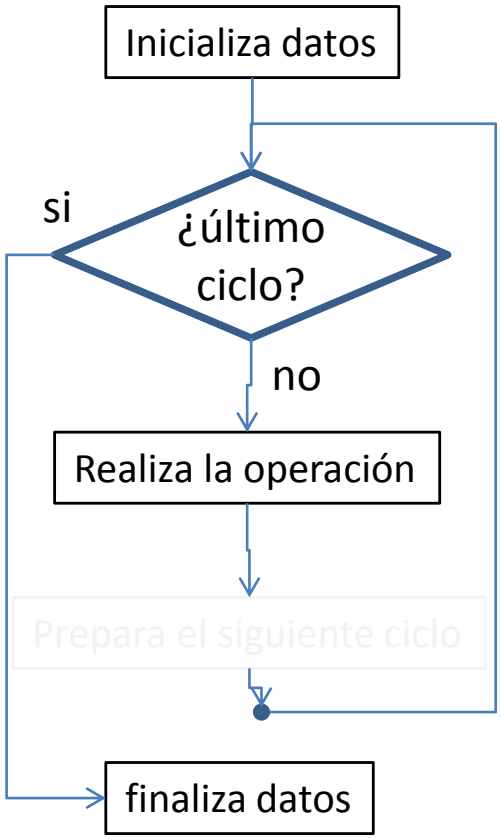

```
!          %o0 : N
!          %o1 :resultado
!          %l1  : i
!          %l2  : j
primo:     mov 2,%l1
ciclo1:    mov 1,%l2
ciclo2:    umul %l1,%l2,%l3
           cmp %l3,%o0
           ble si_1
           nop
no_1:      add %l1,1,%l1
           umul %l1,%l1,%l4
           bg si_2
           nop
no_2:      ba ciclo1
           nop
si_2:      mov 0,%o1
           ba fin
           nop
si_1:      be si_3
           nop
no_3:      add %l2,1,%l2
           ba ciclo2
           nop
si_3:      mov 1,%01
           ba fin
fin:       nop
```

# Residuo con división

Q=A/B
R=A-Q*B

```
                mov 7,%o0       ! %o0 dividendo
                mov 3,%o1       ! %o1 divisor
                                !%2 cociente
                                !%3 residuo
sres:           sdiv %o0,%o1,%o2
                smul %o2,%o1,%l0
                sub %o0,%l0,%o3
fin:            nop
```

# Residuo con restas




```
                mov 7,%o0       ! %o0 dividendo
                mov 3,%o1       ! %o1 divisor
                                ! %o2 cociente
                                ! %o0 residuo
sres:           mov 0,%o2

ciclo:          cmp %o0,%o1
                bl fin_sres
                nop

                sub %o0,%o1,%o0
                add %o2,1,%o2

                ba ciclo
                nop

fin_sres:       nop
```

# Primos

```
            mov 11,%o0          !           %o0 : N
                                !           %o1
:resultado

primo:      mov 2,%l0
            mov %o0,%l2

cicloP:     umul %l0,%l0,%l1
            cmp %l1,%l2
            bg si_pri
            nop

            mov %l2,%o0
            mov %l0,%o1


!  calcula residuos (sres)
```

```
cmp  %o0,0
            be no_pri
            nop

            add %l0,1,%l0
            ba cicloP
            nop

no_pri:     mov 1,%o1
            ba fin
            nop

si_pri:     mov 0,%o1

fin:        ba fin
            nop
```

# MULT_MAT (1/3)

%g1 dir. de $a_0$
%g2 dir. de $b_0$
%g3 dir. de $c_0$
} apuntadores entradas

%g4 $i$
%g5 $j$
%g6 $k$

$$c_{i,j} = \sum_{k=0}^{2} a_{i,k} b_{k,j,}$$

%g7 =$a_{i,k}$ =m(%g1+4*( (3*%g4)+%g6))
%o0 =$b_{k,j}$ =m(%g2+4*( (3*%g6)+%g5))
%o1 =$c_{i,j}$ =m(%g3+4*( (3*%g4)+%g5))
%o2 operaciones

```
MULT:
   !-- para i
        MOV 0,%g4
ciclo_i: CMP %g4,3
        BE fin_i
        NOP
                        !para j
                             MOV 0,%g5
                        ciclo_j: CMP %g5,3
                             BE fin_j
                             NOP
                                  --para k
                                  MOV 0,%g6
                                  ! MOV 0,%o2
                        ciclo_k: CMP %g6,3
                                  BE fin_k
                                  NOP
                                  !  --a_ik
                                  UMUL %g4,3,%o2
                                  ADD %o2,%g6,%o2
                                  UMUL %o2,4,%o2
                                  LD [%g1+%o2],%g7
                                  !  --b_kj
                                  UMUL %g6,3,%o2
                                  ADD %o2,%g5,%o2
                                  UMUL %o2,4,%o2
                                  LD [%g2+%o2],%o0
                                  !  --sum
                                  SMUL %g7,%o0,%o2
                                  ADD %o1,%o2,%o1
        ADD %g4,1,%g4
        BA ciclo_i
        NOP
fin_i:  RETL
        NOP
                             ADD %g5,1,%g5
                             BA ciclo_j
                             NOP
                        fin_j:
                                  ADD %g6,1,%g6
                                  BA ciclo_k
                                  NOP
                                  fin_k: !--c_ij
                                  UMUL %g4,3,%o2
                                  ADD %o2,%g5,%o2
                                    UMUL %o2,4,%o2
                                  ST %o1,[%g3+%o2]
```

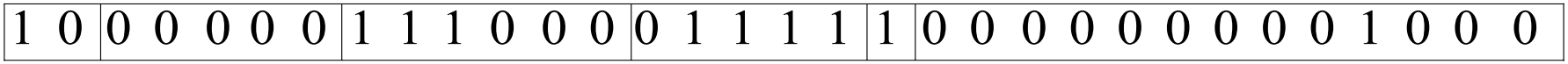
# Llamado a rutina

| 0 1 | cte30 |
|---|---|

| CALL cte30 |
|---|
| %o7<-PC<br>PC<-PC+4*cte30 |

```
 5 00004000 90002005          mov 5,%o0
 6 00004004 40000007          call rutina
 7 00004008 80000000          nop
 8 0000400C 90002009          mov 9,%o0
 9 00004010 40000004          call rutina
10 00004014 80000000          nop
11 00004018 10800000 fin:     ba fin
12 0000401C 80000000          nop
13
14 00004020 92022030 rutina:  add %o0,0x30,%o1
15 00004024 81C3E008          retl
16 00004028 80000000          nop
```

# Retorno de rutina




RETL

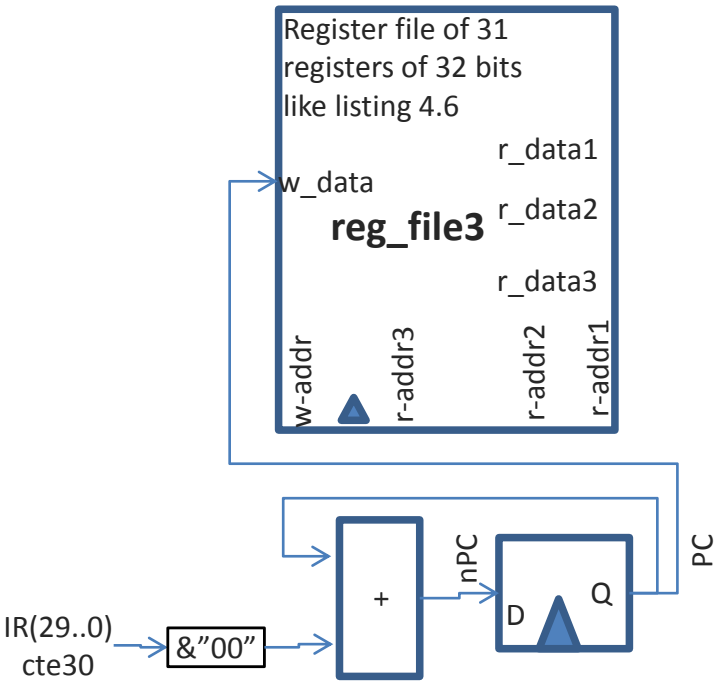
PC<-%o7+8

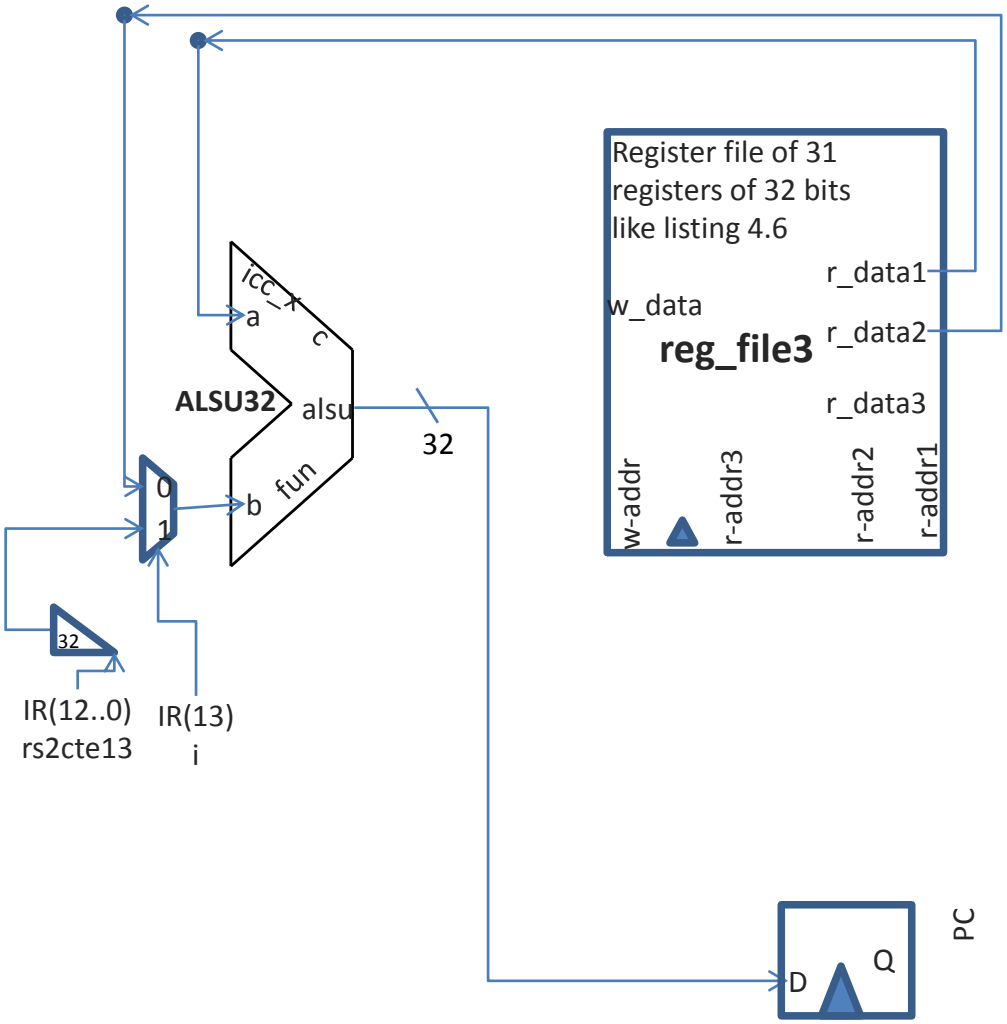
```
 5 00004000 90002005          mov 5,%o0
 6 00004004 40000007          call rutina
 7 00004008 80000000          nop
 8 0000400C 90002009          mov 9,%o0
 9 00004010 40000004          call rutina
10 00004014 80000000          nop
11 00004018 10800000     fin: ba fin
12 0000401C 80000000          nop
13
14 00004020 92022030 rutina: add %o0,0x30,%o1
15 00004024 81C3E008          retl
16 00004028 80000000          nop
```

# Modificación del circuito

| 0 1 | cte30 |
|---|---|

| CALL cte30 |
|---|
| %o7<-PC |
| PC<-PC+4*cte30 |






| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

RETL

PC<-%o7+8

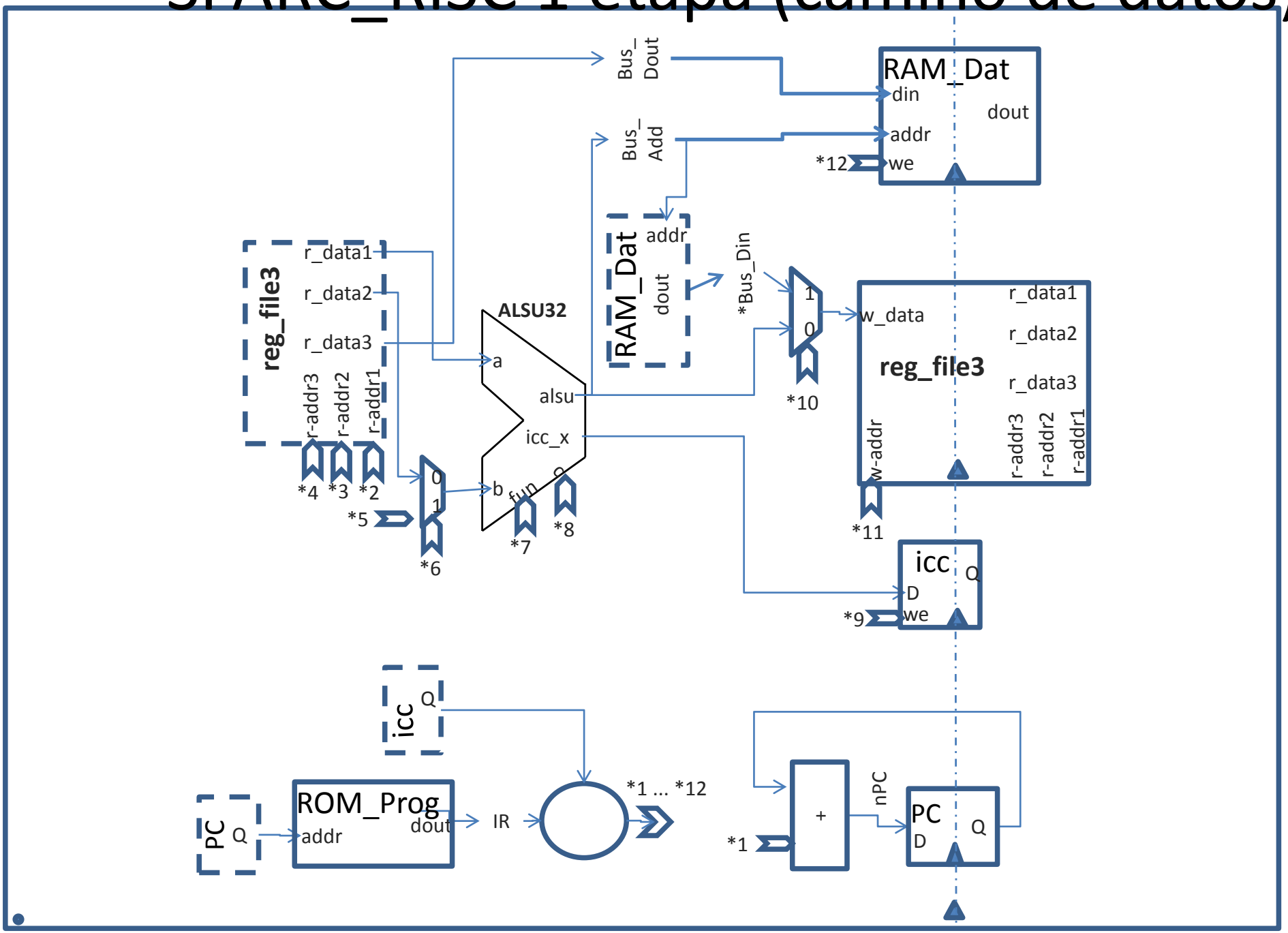
# Llamando la multiplicación de matrices

```
        MOV MA,%g1      !--A
        MOV MB ,%g2     !--B
        MOV ME,%g3      !--E
        CALL MULT       !--E=AB
        NOP
        MOV ME,%g1      !--E
        MOV MC,%g2      !--C
        MOV MD,%g3      !--D
        CALL MULT       !--D=EC
        NOP

FIN:    BA FIN
        NOP

!No se puede usar D en vez de E
!porque después tiene que escribir
!y leer la matriz D, y cuando la escribe
!daña lo que tiene que leer.
```

```
ultimo:
.skip -ultimo
 ! comienza desde cero

MA:
 .word 1
 .word 2
 .word 3
 .word 4
 .word 5
 .word 6
 .word 7
 .word 8
 .word 9
MB:
 .word 10
 .word 11
 .word 12
  ! E. T. C.
```

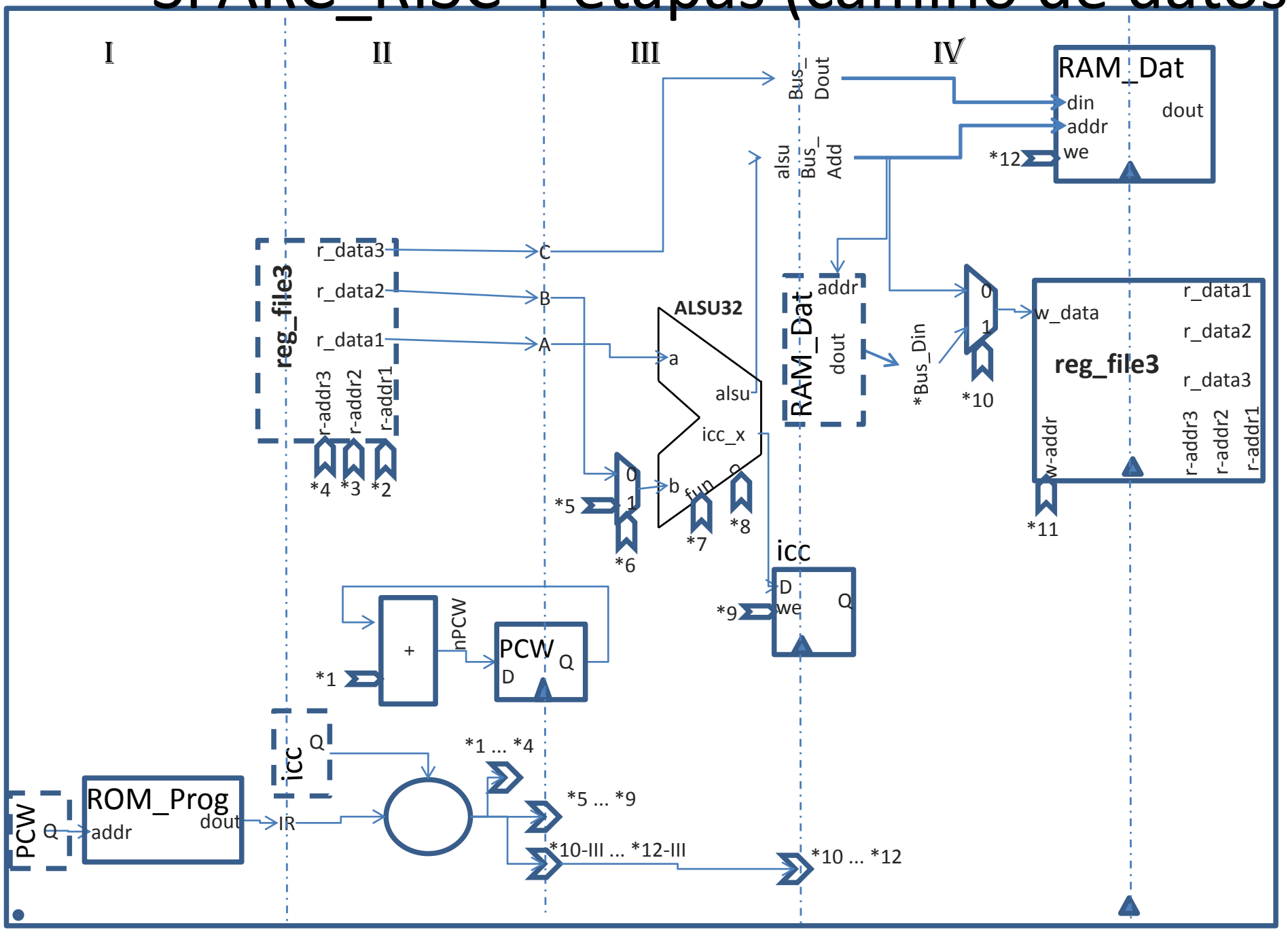
micro_mem

# SPARC_RISC 1 etapa (camino de datos)



micro_mem

# SPARC_RISC 4 etapas (camino de datos



micro_mem

# SPARC_RISC 4 etapas con avance datos