Ejercicio 20 - Guia 7

Enunciado

20. El standard SPARC incluye la instrucción *addxcc (add with carry)* la cual es similar a *addcc* salvo que además de los sumandos estándar suma también el contenido del bit de carry. Se pide:

    a) Dar un ejemplo de su aplicación en el contexto de un programa que realiza la suma de números de 64 bits.

    b) Detalle los cambios que deberían ser introducidos en la memoria de control de un procesador ARC para que *addxcc* forme parte de su set de instrucciones. Para ello considerar que esta instrucción debe cumplir con el formato de operaciones aritmético-lógicas con un op3 igual a *010011*.

ADDXcc — Add with Carry and modify icc

Operation:

r[rd] ← r[rs1] + operand2 + c, where operand2 = (r[rs2] or sign extnd(simm13))
n ← r[rd]<31>
z ← if r[rd] =0 then 1, else 0
v ← (r[rs1]<31> AND operand2<31> AND not r[rd]<31>)
OR (not r[rs1]<31> AND not operand2<31> AND r[rd]<31>)
c ← (r[rs1]<31> AND operand2<31>)
OR (not r[rd]<31> AND (r[rs1]<31> OR operand2<31>))

Assembler Syntax:

addxcc *regrs1, reg_or_imm, regrd*

Description:

ADDXcc adds the contents of r[rs1] to either the contents of r[rs2] if the instruction's *i* bit equals zero, or to a 13-bit, sign-extended immediate operand if *i* equals one. It then adds the PSR's carry bit (*c*) to that result. The final result is placed in the register specified in the *rd* field. ADDXcc also modifies all the integer condition codes in the manner described above.

a)Ejemplo de aplicación

Supongamos que queremos sumar dos números de 64 bits en una arquitectura de 32 bits. Como no se puede hacer directamente, lo hacemos por partes: ab y cd son los dos números divididos en dos partes.

Suponiendo:

%r1→ a

%r2→ b } primer número

%r3→ c

%r4 → d } segundo número

%r5 } resultado

%r6

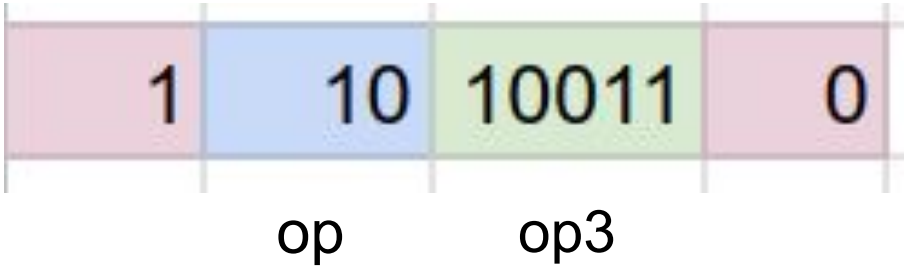
addcc %r2,%r4,%r6 → suma los 32 bits menos significativos primero y cambia los flags

addxcc %r1,%r3,%r5 → suma los 32 bits más significativos y también el carry anterior

b)Agregar ADDXCC a la lista de instrucciones

Tenemos que definir las microinstrucciones necesarias que irán en la ROM para poder ejecutar esta instrucción SPARC en nuestra arquitectura ARC.

Primero calculamos la **dirección** donde empezará el set de microinstrucciones, sabiendo el op3.



```
1612: R[temp0] = ADD[R[r0],R[r0]]
1613: IF C=1 then GOTO 1615;  //Si hubo un carry anterior salta
1614: GOTO 1616; //No hubo un carry anterior
1615: R[temp0] = INC[R[temp0]];  //Sumo el carry en temp0
1616: IF R[IR[13]] then GOTO 1618; //Si está en modo inmediato salta
1617: R[rd] = ADDCC[R[rs1],R[rs2]];  // Suma
      GOTO 1620;
1618: R[temp1] = SEXT13[R[IR]]; //Extiende el signo del campo simm13
1619: R[rd] = ADDCC[R[rs1],R[temp1]]; //Suma
1620: R[rd] = ADDCC[R[rd],R[temp0]] //Suma el carry
      GOTO 2047
```