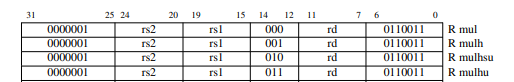


Instrucciones de multiplicación:



Procesos por instrucción:

**AUIPC**

**Instrucción tipo U** Opcode 0010111

Escribe en el registro con la dirección *rd,* la suma del pc más el dato almacenado en *immediate*, desplazado 12 bits a la izquierda y rellenado con ceros



Registers[IR(7:11)](0:20)<-(IR(12:31) + PC)

Registers[IR(7:11)](20:31)<-(0)

**LUI**

**Instrucción tipo U** Opcode 0110111

Escribe en el registro con la dirección *rd* el dato almacenado en *immediate*, desplazado 12 bits a la izquierda y rellenado con ceros



Registers[IR(7:11)](0:20)<-(IR(12:31))

Registers[IR(7:11)](20:31)<-(0)

**JAL**

**Instrucción tipo J** Opcode 1101111

Escribe en el registro con la dirección *rd,* la dirección de la próxima instrucción (se le suma 4 al pc actual), luego almacena en el PC la suma de este mas el dato almacenado en *offset* extendido en signo, (es decir, los bits del 21 al 31 se rellenan con el bit más significativo de *offset)* con el siguiente orden desde el LSB al MSB : IR[21:30 ; 20; 12:19; 31]



Registers[ IR (7:11) ] <- (pc+4)

Pc<- Pc+ sext(IR (12:31))-- sign extended IR

**JALR**

**Instrucción tipo J** Opcode 1100111

Escribe en el registro con la dirección *rd,* la dirección de la próxima instrucción (se le suma 4 al pc actual), luego almacena en el PC la suma del contenido del registro con la dirección rs1 más el dato almacenado en *offset* extendido en signo, (es decir, los bits del 12 al 31 se rellenan con el bit más significativo de *offset).*



Registers[IR(7:11)] <- PC+4;

Pc<- ( Registers[IR(15:19)]+ sext(IR(20:31)) )

**BEQ**

**Instrucción tipo B** OpCode 1100011 / 000 (Esta segunda parte del opcode, corresponde a los bits 12:14 del IR)

Compara el contenido del registro con dirección rs1 con el contenido del registro con dirección rs2, si son iguales, suma al pc actual el valor del *offset* en signo extendido (los bits del 13 al 31 del offset se rellenan con el bit 12 del *offset*), almacenado en el *IR* de la siguiente forma, desde LSB a MSB: IR[8:11 ; 25:30 ; 7 ; 31 ] Si el contenido de los registros es diferente, no hace nada.



If ( Registers[IR (20:24)] == Registers[IR (15:19)] )

Pc= + Pc + sext (offset)-- Desordenado según instrucción tipo B

**BNE**

**Instrucción tipo B** OpCode 1100011 / 001

Compara el contenido del registro con dirección rs1 con el contenido del registro con dirección rs2, si son diferentes, suma al pc actual el valor del *offset* en signo extendido (los bits del 13 al 31 del offset se rellenan con el bit 12 del *offset*), almacenado en el *IR* de la siguiente forma, desde LSB a MSB: IR[8:11 ; 25:30 ; 7 ; 31 ] Si el contenido de los registros es igual, no hace nada.



If ( Registers[IR (20:24)] != Registers[IR (15:19)] )

Pc<- Pc + sext (offset)-- Ordenado según instrucción tipo B

**BLT**

**Instrucción tipo B** Opcode 1100011 / 100

Compara el contenido del registro con dirección rs1 con el contenido del registro con dirección rs2, si el contenido del registro con dirección rs1 es menor al contenido del registro con dirección rs2, (el contenido de ambos registros se trata como números de complemento a dos) suma al pc actual el valor del *offset* en signo extendido (los bits del 13 al 31 del offset se rellenan con el bit 12 del *offset*), almacenado en el *IR* de la siguiente forma, desde LSB a MSB: IR[8:11 ; 25:30 ; 7 ; 31 ] Si no es menor, no hace nada.



If( Registers[IR (15:19)] < Registers[IR (20:24)] )

Pc<- Pc + sext (offset) -- Ordenado según instrucción tipo B

**BGE**

**Instrucción tipo B** Opcode 1100011 / 101

Compara el contenido del registro con dirección rs1 con el contenido del registro con dirección rs2, si el contenido del registro con dirección rs1 es mayor al contenido del registro con dirección rs2, (el contenido de ambos registros se trata como números de complemento a dos) suma al pc actual el valor del *offset* en signo extendido (los bits del 13 al 31 del offset se rellenan con el bit 12 del *offset*), almacenado en el *IR* de la siguiente forma, desde LSB a MSB: IR[8:11 ; 25:30 ; 7 ; 31 ] Si no es mayor, no hace nada.



If( Registers[IR (15:19)] >= Registers[IR (20:24)] )

Pc<- Pc + sext (offset) -- Ordenado según instrucción tipo B

**BLTU**

**Instrucción tipo B** Opcode 1100011 / 110

Compara el contenido del registro con dirección rs1 con el contenido del registro con dirección rs2, si el contenido del registro con dirección rs1 es menor al contenido del registro con dirección rs2, (el contenido de ambos registros se trata como números sin signo) suma al pc actual el valor del *offset* en signo extendido (los bits del 13 al 31 del offset se rellenan con el bit 12 del *offset*), almacenado en el *IR* de la siguiente forma, desde LSB a MSB: IR[8:11 ; 25:30 ; 7 ; 31 ] Si no es menor, no hace nada.



If( Registers[IR (15:19)] < Registers[IR (20:24)] )

Pc<- Pc + sext (offset) -- Desordenado según instrucción tipo B

**BGEU**

**Instrucción tipo B** Opcode 1100011 / 111

Compara el contenido del registro con dirección rs1 con el contenido del registro con dirección rs2, si el contenido del registro con dirección rs1 es mayor al contenido del registro con dirección rs2, (el contenido de ambos registros se trata como números sin signo) suma al pc actual el valor del *offset* en signo extendido (los bits del 13 al 31 del offset se rellenan con el bit 12 del *offset*), almacenado en el *IR* de la siguiente forma, desde LSB a MSB: IR[8:11 ; 25:30 ; 7 ; 31 ] Si no es mayor, no hace nada.



If( Registers[IR (15:19)] >= Registers[IR (20:24)] )

Pc<- Pc + sext (offset) -- Desordenado según instrucción tipo B

**LB**

**Instrucción tipo I** Opcode 0000011 / 000

Carga un byte de memoria extendido en signo desde la dirección obtenida por la suma de offset y el contenido almacenado en el registro con dirección rs1. Escribe el dato cargado en memoria en el registro con dirección rd.



I/O<-Memory[ Registers[ IR (15:19) ] ]

Registers[IR (7:11) ]<- sext ( I/O +sext( IR( 20: 31 ) ) )

**LH**

**Instrucción tipo I** Opcode 0000011 / 001

Carga dos bytes de memoria extendidos en signo desde la dirección obtenida por la suma de offset(tambien extendido en signo) y el contenido almacenado en el registro con dirección rs1. Escribe el dato cargado en memoria en el registro con dirección rd.



I/O<-Memory[ Registers[ IR (15:19) ] ]-- carga 2 byte

Registers[IR (7:11) ]<- sext ( I/O +sext( IR( 20: 31 ) ) )--

**LW**

**Instrucción tipo I** Opcode 0000011 / 010

Carga cuatro bytes de memoria extendidos en signo desde la dirección obtenida por la suma de offset(tambien extendido en signo) y el contenido almacenado en el registro con dirección rs1. Escribe el dato cargado en memoria en el registro con dirección rd.



I/O<-Memory[ Registers[ IR (15:19) ] ]-- carga 4 byte

Registers[IR (7:11) ]<- sext ( I/O +sext( IR( 20: 31 ) ) )--

**LBU**

**Instrucción tipo I** Opcode 0000011 / 100

Carga un byte de memoria extendido con ceros, desde la dirección obtenida por la suma de offset y el contenido almacenado en el registro con dirección rs1. Escribe el dato cargado en memoria en el registro con dirección rd.



I/O<-Memory[ Registers[ IR (15:19) ] ]-- Como hago la address para memory?? Carga 1 byte

Registers[IR (7:11) ]<- ( I/O +sext( IR( 20: 31 ) ) )

**LHU**

**Instrucción tipo I** Opcode 0000011 / 101

Carga dos bytes de memoria extendidos con ceros desde la dirección obtenida por la suma de offset(tambien extendido en signo) y el contenido almacenado en el registro con dirección rs1. Escribe el dato cargado en memoria en el registro con dirección rd.



I/O<-Memory[ Registers[ IR (15:19) ] ]-- carga 2 byte

Registers[IR (7:11) ]<- ( I/O +sext( IR( 20: 31 ) ) )--

**SB**

**Instrucción tipo S** Opcode 0100011 / 000

Guarda en memoria el byte menos significativo del registro con dirección rs2, en la dirección de memoria obtenida por la suma de offset y el contenido almacenado en el registro con dirección rs1. El offset se encuentra en los bits 7 a 11 del IR y continua en los bits25 a 31 con longitud de 12 bits, siendo el bit 7 el LSB.



Memory[IR(15:19+sext(offset))]<- LSB(Registers[IR(20:24)])-- Offset de acuerdo a tipo S

**SH**

**Instrucción tipo S** Opcode 0100011 / 001

Guarda en memoria los dos bits menos significativos del registro con dirección rs2, en la dirección de memoria obtenida por la suma de offset y el contenido almacenado en el registro con dirección rs1. El offset se encuentra en los bits 7 a 11 del IR y continua en los bits25 a 31 con longitud de 12 bits, siendo el bit 7 el LSB.



Memory[IR(15:19+sext(offset))]<- 2LSB(Registers[IR(20:24)])-- Offset de acuerdo a tipo S

**SW**

**Instrucción tipo S** Opcode 0100011 / 010

Guarda en memoria los cuatro bits del registro con dirección rs2, en la dirección de memoria obtenida por la suma de offset y el contenido almacenado en el registro con dirección rs1. El offset se encuentra en los bits 7 a 11 del IR y continua en los bits25 a 31 con longitud de 12 bits, siendo el bit 7 el LSB.



Memory[IR(15:19+sext(offset))]<- Registers[IR(20:24)]-- Offset de acuerdo a tipo S

**ADDI**

**Instrucción tipo I** Opcode 0010011 / 000

Suma el contenido del registro con dirección rs1 con el dato immediate extendido en signo, almacenado en los bits IR[20:31] y almacena el resultado en el registro con dirección rd.

**Nota: esta operación ignora el overflow aritmético.**



Registers[ IR(7:11) ]<- Registers[ IR(15:19) ] + IR(20:30)

**SLTI**

**Instrucción tipo I** Opcode 0010011 / 010

Compara el contenido del registro con dirección rs1 con el dato immediate extendido en signo, almacenado en los bits IR[20:31], como números de complemento a dos. Si el contenido del registro con direccion rs1 es menor, escribe 1 en el registro con dirección rd, de lo contrario escribe 0 en este registro.



If ( Registers[ IR(15:19) ] < IR(20:31) ) Registers[ IR(7:11) ] <- 1—Numeros 2sC

Else Registers[ IR(7:11) ] <- 0

**Nota para desarrollo: si es menor se puede encender una bandera del SR asi que se puede asignar ese sr**

**SLTIU**

**Instrucción tipo I** Opcode 0010011 / 011

Compara el contenido del registro con dirección rs1 con el dato immediate extendido en signo, almacenado en los bits IR[20:31], como números sin signo. Si el contenido del registro con direccion rs1 es menor, escribe 1 en el registro con dirección rd, de lo contrario escribe 0 en este registro.



If ( Registers[ IR(15:19) ] < IR(20:31) ) Registers[ IR(7:11) ] <- 1 – Numeros unsigned

Else Registers[ IR(7:11) ] <- 0

**XORI**

**Instrucción tipo I** Opcode 0010011 / 100

Calcula el XOR a nivel de bits entre el registro con dirección rs1 con el dato immediate extendido en signo, almacenado en los bits IR[20:31], como números sin signo. Escribe el resultado en el registro con dirección rd.



Registers [ IR(7:11) ] <- IR ( 20:31 ) XOR Registers [ IR(15:19) ]

**ORI**

**Instrucción tipo I** Opcode 0010011 / 110

Calcula el OR a nivel de bits entre el registro con dirección rs1 con el dato immediate extendido en signo, almacenado en los bits IR[20:31], como números sin signo. Escribe el resultado en el registro con dirección rd.



Registers [ IR(7:11) ] <- IR ( 20:31 ) OR Registers [ IR(15:19) ]

**ANDI**

**Instrucción tipo I** Opcode 0010011 / 011

Calcula el AND a nivel de bits entre el registro con dirección rs1 con el dato immediate extendido en signo, almacenado en los bits IR[20:31], como números sin signo. Escribe el resultado en el registro con dirección rd.



Registers [ IR(7:11) ] <- IR ( 20:31 ) OR Registers [ IR(15:19) ]

**SLLI**

**Instrucción tipo I** Opcode 0010011 / 001

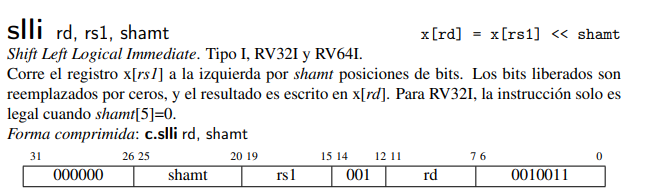
Mueve el contenido del registro con direccion rs1 a la izquierda el número de bits (igual a la cantidad de desplazamientos a la izquierda) almacenado en shamt, en IR [20:25], se añaden ceros al desplazar. Escribe el resultado en el registro con dirección rd.

**Nota: la instrucción solo es valida si shamt[5]=IR[25]=0**



Rg[ IR(15:19) ]<< --IR(20:25) cantidad de veces, reemplaza con 0

Rg[ IR(7:11) ] <- Rg[ IR(15:19) ]



**Nota para desarrollo: esta instrucción requiere un contador descendente**

**Nota: IR(25) DEBE ser 0**

**SRLI**

**Instrucción tipo I** Opcode 0010011 / 101 / 0000000 , este último opcode son los bits 25:30 del IR

**Nota para desarrollo: el bit relevante en el ultimo opcode es el bit 30 el IR**

Mueve el contenido del registro con direccion rs1 a la derecha el número de bits (igual a la cantidad de desplazamientos a la derecha) almacenado en shamt, en IR [20:25], se añaden ceros al desplazar. Escribe el resultado en el registro con dirección rd.

**Nota: la instrucción solo es valida si shamt[5]=IR[25]=0**



Rg[ IR(15:19) ] >> --IR(20:25) cantidad de veces, reemplaza con 0

Rg[ IR(7:11) ] <- Rg[ IR(15:19) ]

**Nota: esta instrucción requiere un contador descendente**

**SRAI**

**Instrucción tipo I** Opcode 0010011 / 011 / 010000 – este ultimo bit 25:30 del IR

**Nota para desarrollo: el bit relevante en el ultimo opcode es el bit 30 el IR**

Mueve el contenido del registro con direccion rs1 a la derecha el número de bits (igual a la cantidad de desplazamientos a la derecha) almacenado en shamt, en IR [20:25], se añaden copias del MSB del dato al desplazar. Escribe el resultado en el registro con dirección rd.

**Nota: la instrucción solo es valida si shamt[5]=IR[25]=0**



Rg[ IR(15:19) ] >> --IR(20:25) cantidad de veces , reemplaza con el MSB

Rg[ IR(7:11) ] <- Rg[ IR(15:19) ]

**ADD**

**Instrucción tipo R** Opcode 0110011 / 000 / 0000000 Este ultimo opcode son bits del 25 al 31 del IR

Suma el contenido del registro con dirección rs2 al contenido del registro con dirección rs1 y el resultado lo almacena en el registro con dirección rd.

**Nota**: esta instrucción ignora el overflow aritmético.



Rg[ IR (7:11) ] 7:11= Rg[ IR (15:19) ] - Rg[ IR (20:24) ]

**SUB**

**Instrucción tipo R** Opcode 0110011 / 000 / 0100000 Este opcode se diferencia de ADD solo por el bit 30 del IR

Resta el contenido del registro con direccion rs2 al contenido del registro con direccion rs1 y el resultado lo almacena en el registro con direccion rd.



Rg[ IR (7:11) ] 7:11= Rg[ IR (15:19) ] - Rg[ IR (20:24) ]

**SLL**

**Instrucción tipo R** Opcode: 0110011 / 001

Mueve el contenido del registro con dirección rs1 a la izquierda, la cantidad de veces almacenada en el registro con dirección rs2 (se toman solo los 5 bits menos significativos), llenando los bits con 0, el resultado es escrito en el registro con direccion rd.



Rg[ IR (15:19) ] << -- { Rg[ IR(20:24) ] (0:5) } veces, rellena con 0

Rg[IR (7:11) ]<- Rg[ IR (15:19) ]

**SLT**

**Instrucción tipo R** Opcode: 0110011 / 010

Compara el contenido del registro con dirección rs1 con el contenido del registro con dirección rs2 (trata ambos numeros, como numeros con signo). Si el contenido del registro con dirección rs1 es menor, escribe en el registro con dirección rd 1, de lo contrario escribe 0. Tanto el contenido del registro



If ( Registers[ IR(15:19) ] < Registers[ IR(20:24) ] ) Registers[ IR(7:11) ] <- 1 – Numeros unsigned

Else Registers[ IR(7:11) ] <- 0

**SLTU**

**Instrucción tipo R** Opcode: 0110011 / 011

Compara el contenido del registro con dirección rs1 con el contenido del registro con dirección rs2. (trata ambos numeros, como numeros sin signo) Si el contenido del registro con dirección rs1 es menor, escribe en el registro con dirección rd 1, de lo contrario escribe 0.

oki

If ( Registers[ IR(15:19) ] < Registers[ IR(20:24) ] ) Registers[ IR(7:11) ] <- 1 – Numeros unsigned

Else Registers[ IR(7:11) ] <- 0

**XOR**

**Instrucción tipo R** Opcode: 0110011 / 100

Calcula el XOR bit a bit entre el contenido de los registros con dirección rs1 y rs2, el resultado lo escribe en el registro con dirección rd.



Rg[ IR (7:11) ] 7:11= Rg[ IR (15:19) ] XOR Rg[ IR (20:24) ]

**SRL**

**Instrucción tipo R** Opcode: 0110011 / 101 / 0000000 se diferencia de sra por el bit 30 del IR

Mueve el contenido del registro con dirección rs1 a la derecha, la cantidad de veces almacenada en el registro con dirección rs2 (se toman solo los 5 bits menos significativos), llenando los bits con 0, el resultado es escrito en el registro con dirección rd.



Rg[ IR (15:19) ] >> -- { Rg[ IR(20:24) ] (0:5) } veces, rellena con 0

Rg[IR (7:11) ]<- Rg[ IR (15:19) ]

**SRA**

**Instrucción tipo R** Opcode: 0110011 / 101 / 0100000

Mueve el contenido del registro con dirección rs1 a la derecha, la cantidad de veces almacenada en el registro con dirección rs2 (se toman solo los 5 bits menos significativos), llenando los bits con el bit más significativo, el resultado es escrito en el registro con dirección rd.



Rg[ IR (15:19) ] >> -- { Rg[ IR(20:24) ] (0:5) } veces, rellena con el MSB de Rg[ IR (15:19) ]

Rg[IR (7:11) ]<- Rg[ IR (15:19) ]

**OR**

**Instrucción tipo R** Opcode: 0110011 / 110

Calcula el OR bit a bit entre el contenido de los registros con dirección rs1 y rs2, el resultado lo escribe en el registro con dirección rd.



Rg[ IR (7:11) ] 7:11= Rg[ IR (15:19) ] OR Rg[ IR (20:24) ]

**AND**

**Instrucción tipo R** Opcode: 0110011 / 111

Calcula el AND bit a bit entre el contenido de los registros con dirección rs1 y rs2, el resultado lo escribe en el registro con dirección rd.



Rg[ IR (7:11) ] 7:11= Rg[ IR (15:19) ] AND Rg[ IR (20:24) ]

**CSRRW**

**Instrucción tipo I** Opcode 1110011 / 001

Primero escribe el contenido del Control And Status Register de la dirección csr(bits del 20 al 31 del IR) en el registro con dirección rd, luego escribe el contenido del registro con dirección rs1 en el Control And Status Register con dirección csr.



Rg[ IR(7:11) ] <- CSRs [ IR(20:31) ]

CSRs [ IR(20:31) ] <- Rg[ IR(15:19) ]

**CSRRS**

**Instrucción tipo I** Opcode 1110011 / 001

Almacena el valor del CSRs en la posición csr (bits del 20 al 31 del IR) en el registro con dirección rd, escribe el OR entre este valor y el contenido del registro con direccion rs1 en el CSR con dirección csr.



Rg[ IR(7:11) ] <- CSR [ IR(20:31) ]

CSR [ IR(20:31) ] <- Rg[ IR(15:19) ] OR Rg[ IR(7:11) ]

**CSRRC**

**Instrucción tipo I** Opcode 1110011 / 011

Primero escribe el contenido del Control And Status Register de la dirección csr(bits del 20 al 31 del IR) en el registro con dirección rd, luego escribe el AND entre este valor y el contenido del registro con direccion rs1 en el CSR con dirección csr.



Rg[ IR(7:11) ] <- CSR [ IR(20:31) ]

CSR [ IR(20:31) ] <- Rg[ IR(15:19) ] OR Rg[ IR(7:11) ]

**CSRRWI**

**Instrucción tipo I** Opcode 1110011 / 011

Primero escribe el contenido del Control And Status Register de la dirección csr(bits del 20 al 31 del IR) en el registro con dirección rd, luego escribe el dato zimm IR[15:19] extendido en ceros en el CSR con dirección csr.



Rg[ IR(7:11) ] <- CSR [ IR(20:31) ]

CSR [ IR(20:31) ] <- IR(15:19) -- cero extendido

**CSRRSI**

**Instrucción tipo I** Opcode 1110011 / 110

Primero escribe el contenido del Control And Status Register de la dirección csr(bits del 20 al 31 del IR) en el registro con dirección rd, luego escribe el OR entre este valor y el dato zimm IR[15:19] extendido en ceros en el CSR con dirección csr.



Rg[ IR(7:11) ] <- CSR [ IR(20:31) ]

CSR [ IR(20:31) ] <- IR(15:19) OR Rg[ IR(7:11) ]

**CSRRCI**

**Instrucción tipo I** Opcode 1110011 / 111

Primero escribe el contenido del Control And Status Register de la dirección csr(bits del 20 al 31 del IR) en el registro con dirección rd, luego escribe el AND entre este valor y el dato zimm IR[15:19] extendido en ceros en el CSR con dirección csr.

Nota: el and se realiza solo en los primeros 5bits del CSR[csa] bedio a que el resto seria una mascara que borra todo.



Rg[ IR(7:11) ] <- CSR [ IR(20:31) ]

CSR [ IR(20:31) ] <- 1sC( IR(15:19)) AND Rg[ IR(7:11) ]

**MUL**

**Instrucción tipo R** Opcode 0110011 000

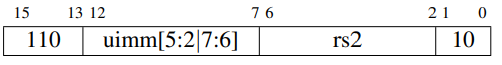
Multiplica el contenido del registro con dirección rs1 y rs2. Almacena el resultado de la multiplicación en el registro con dirección rd.



PUSH Y POP subidas después de corrección.

**SWSP**

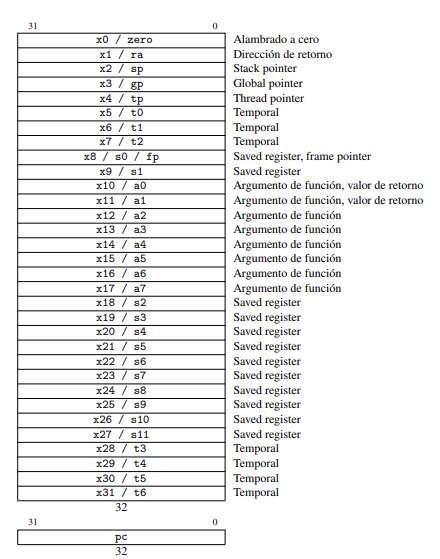
Almacena una palabra en la memoria en la dirección del SP + uimm11



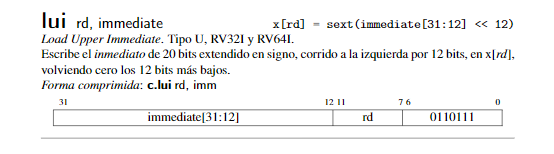
**LWSP**

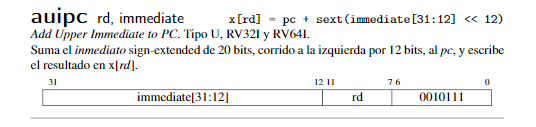
Carga la palabra en el registro con dirección rd, desde la memoria con dirección sp + uimm



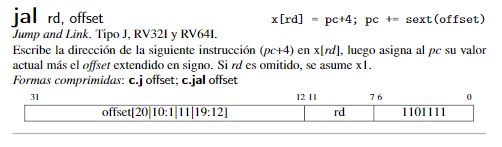


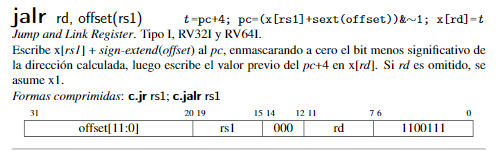
Instrucciones tipo U:



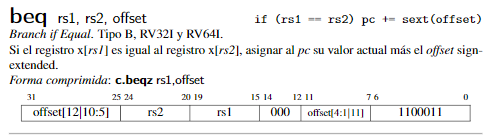


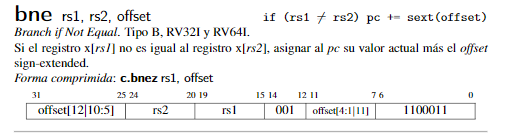
Instrucciones tipo J:

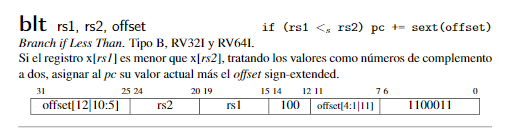


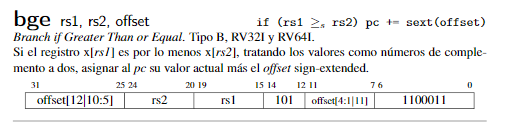


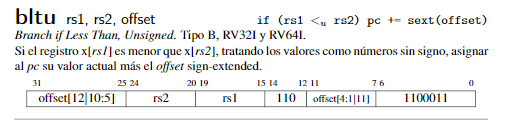
Instrucciones tipo B:

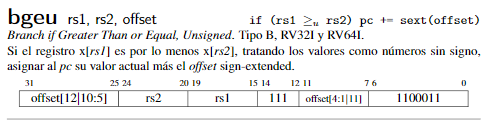




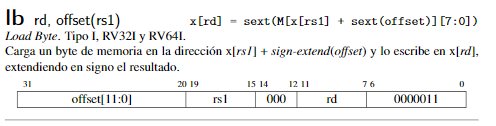


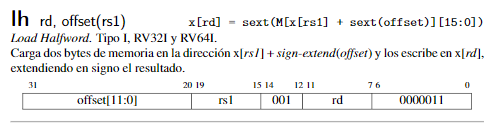


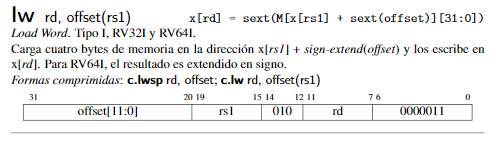


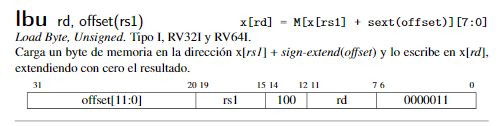


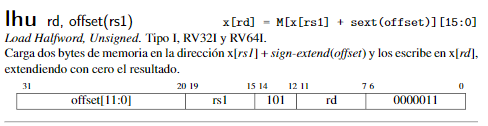
Instrucciones tipo I:

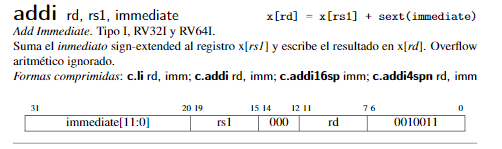


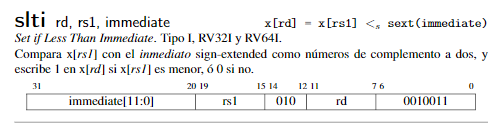


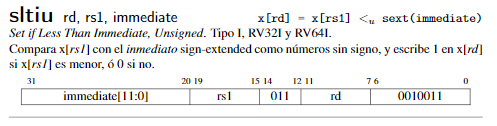


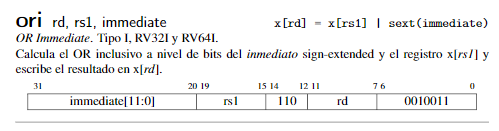
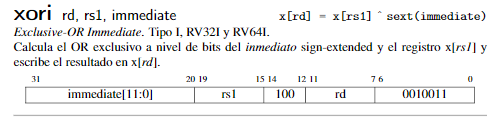


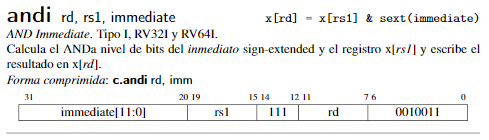


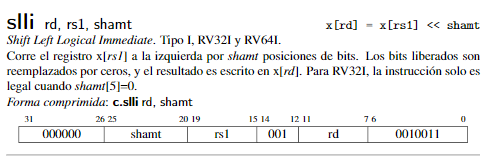


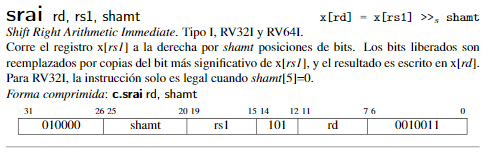
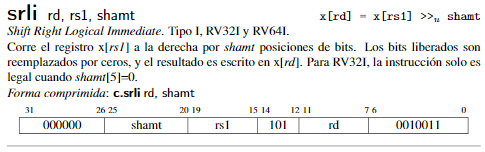


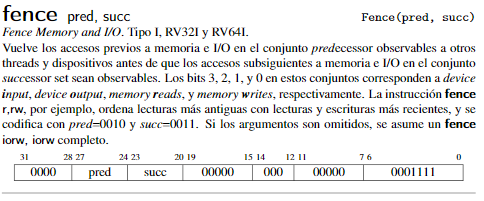


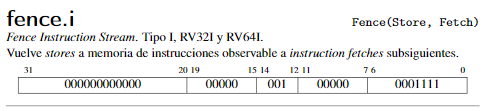


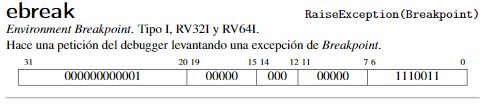
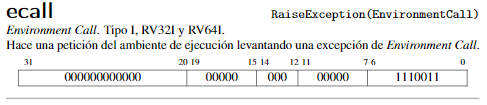


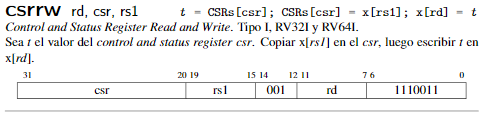


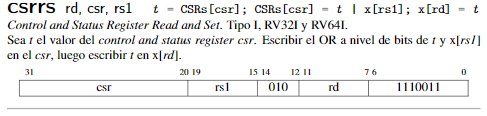


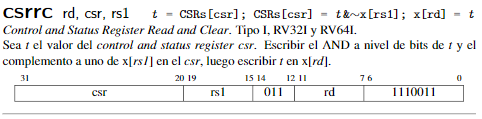


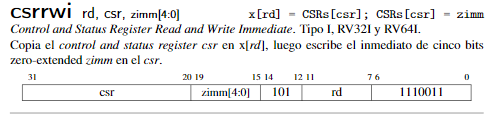


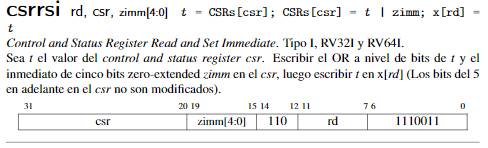


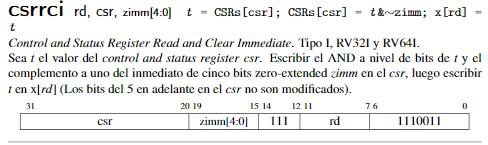




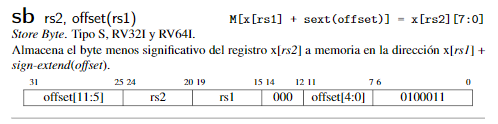


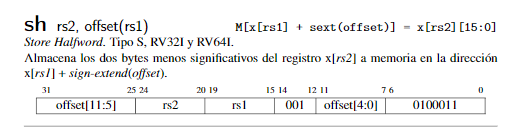


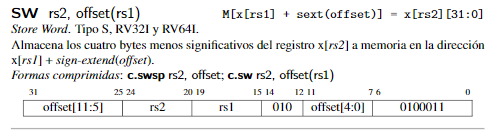




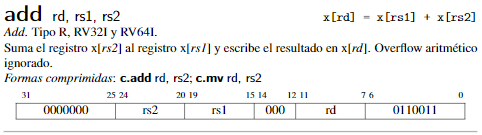
Instrucciones tipo S:

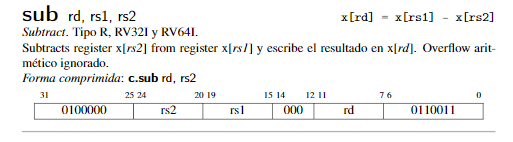


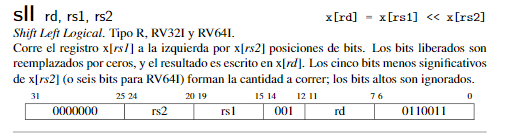


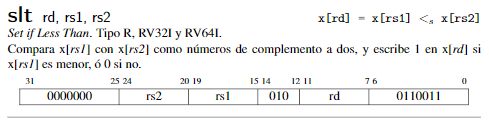


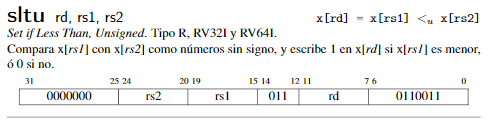
Instrucciones tipo R:

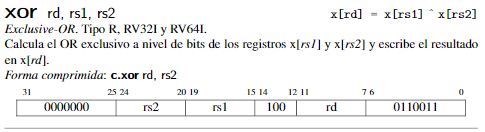


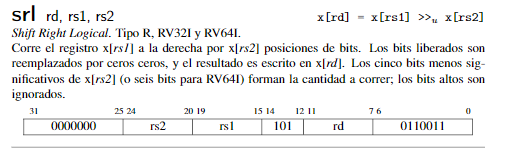


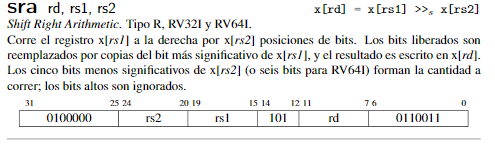


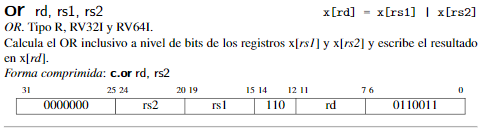


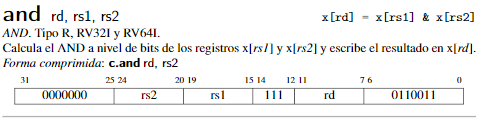












Instrucciones multiplicación:

Instrucciones Tipo R

