

Instituto Tecnológico de Costa Rica
Área Académica de Ingeniería en Computadores
(Computer Engineering Academic Area)

Programa de Licenciatura en Ingeniería en Computadores
(Licentiate Degree Program in Computer Engineering)

Curso: CE-4301 Arquitectura de Computadores I
(Course: CE-4301 Computer Architecture I)



Documento de Descripción del ISA
(First Project)

Realizado por:

Made by:

Fabián Astorga Cerdas 2014040808

Javier Sancho Marín 2014159997

Óscar Ulate Alpízar 201229559

Profesor:

(Professor)

Fabián Zamora Ramírez

Fecha: Cartago, 9 octubre, 2017

(Date: Cartago, October 9, 2017)

1.ASPECTOS GENERALES

Elementos del ISA	JOF32
Clase de ISA	<ul style="list-style-type: none">• RISC• Load/Store
Registros	<ul style="list-style-type: none">• El procesador cuenta con 16 registros de 32 bits en total de los cuales: 14 son de uso general, 1 registro para almacenar el número cero, 1 registro para la bandera de salto
Tipos de datos	<ul style="list-style-type: none">• Bytes de 8 bits• Palabras de 4 bytes
Modos de direccionamiento	<ul style="list-style-type: none">• Inmediato con desplazamiento $[Rn, \#imm]$• Registro $[Rn]$• Registro escalado con desplazamiento $[Rn, \pm Rm, shift]$
Organización de memoria	<ul style="list-style-type: none">• Soporta Big-Endian.• Espacio de direccionamiento: 2MB• Addressability: 32 bits.• Todos los accesos a memoria deben estar alineados.• Los accesos a memoria pueden ser a nivel de byte y palabra.
Formatos de instrucción	<ul style="list-style-type: none">• Data processing immediate shift• Data processing register shift• Data processing immediate• Load/Store Immediate offset• Load/Store register offset• Load/Store multiple• Branch/Branch with link

Operaciones	<ul style="list-style-type: none">• Load/Store.• Operaciones de ALU (lógicas y aritméticas).• Branches.• Saltos.
Codificación	<ul style="list-style-type: none">• Instrucciones de tamaño fijo: 32 bits• El modo de direccionamiento se guarda en el opcode

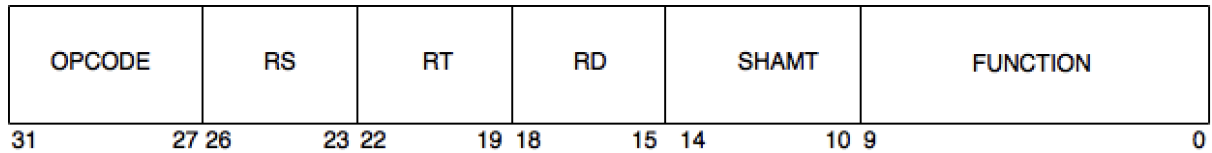
2.NOTACIÓN

Tabla 2A. Convención notacional

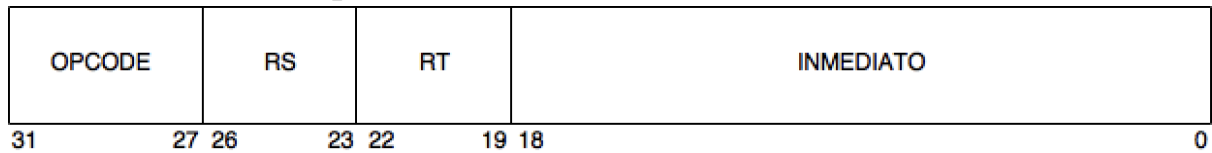
NOTACIÓN	SIGNIFICADO
<i>BaseR</i>	Registro base. Utilizado para dar referencia a un registro que se esté utilizado para llevar a cabo alguna operación. (R0,...,R15)
<i>A[i:d]</i>	Utilizado para delimitar una sección específica de un dato compuesto por un conjunto de bits. Por ejemplo: el código de operación (<i>opcode</i>) puede ser interpretado como la sección del dato que está entre el bit 31 (i) y el bit 26 (d), por lo tanto: <i>opcode = instr[31:26]</i>
<i>#Numero</i>	Número en notación decimal.
<i>*</i>	Operador utilizado para interpretar una multiplicación.
<i>::</i>	Operador utilizado para concatenar dos conjuntos compuestos por uno o más bits.
<i>Offset</i>	Dato compuesto por 19 bits con el objetivo de realizar una operación específica dentro de las instrucciones de formato I. (Véase la sección 4)
<i>PC</i>	Contador de programa. Compuesto por 10 bits, el cual contiene una dirección de memoria de la siguiente instrucción que se debería ejecutar.

3.FORMATO DE INSTRUCCIONES

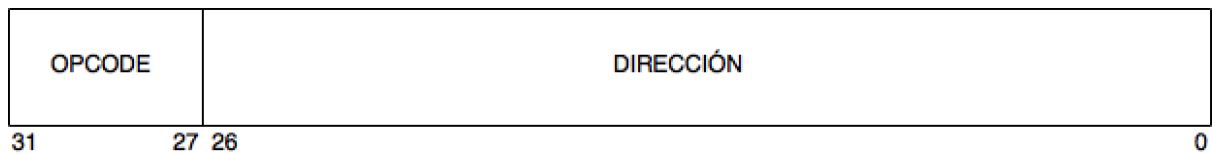
a. Tipo R



b. Tipo I



c. Tipo J



4.SET DE INSTRUCCIONES

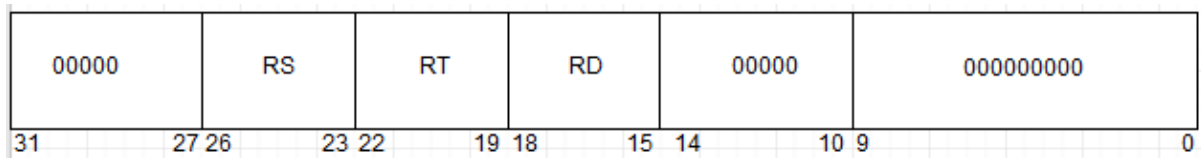
ADD

Suma

FORMATO ENSAMBLADOR

ADD RD, RS, RT

CODIFICACIÓN



OPERACIÓN

if (bit[31:27] == 00000)

RD = RS+RT

DESCRIPCIÓN

En caso de que el código de operación de la instrucción sea el establecido para la suma, se almacena el resultado de dicha suma de los registros RS y RT en el registro RD.

EJEMPLO

ADD R5, R7, R9 : R5 = R7+R9

ADD R2, R2, R4 : R2 = R2+R4

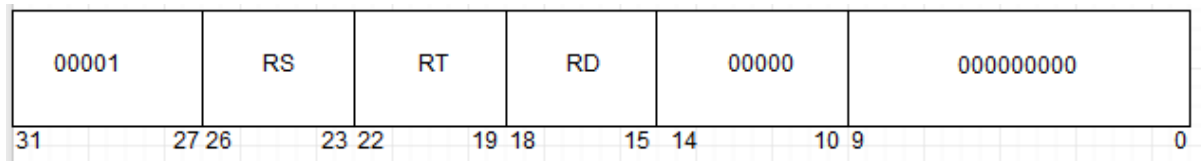
SUB

Resta

FORMATO ENSAMBLADOR

SUB RD, RS, RT

CODIFICACIÓN



OPERACIÓN

if (bit[31:27] == 00001)

RD = RS-RT

DESCRIPCIÓN

En caso de que el código de operación de la instrucción sea el establecido para la resta, se almacena el resultado de dicha resta entre los registros RS y RT en el registro RD.

EJEMPLO

SUB R5, R7, R9 : R5 = R7-R9

SUB R2, R2, R4 : R2 = R2-R4

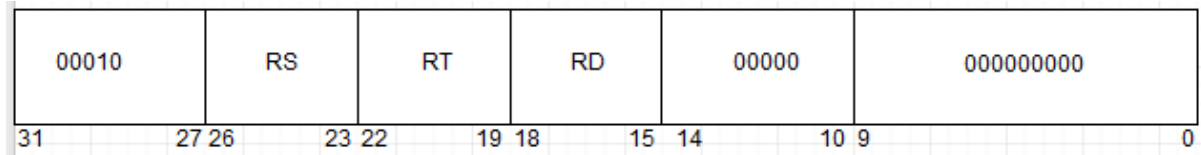
AND

And lógico

FORMATO ENSAMBLADOR

AND RD, RS, RT

CODIFICACIÓN



OPERACIÓN

if (bit[31:27] == 00010)

RD = RS & RT

DESCRIPCIÓN

En caso de que el código de operación indique un and, Aplica este operador lógico entre los registros RS y RT y guarda el resultado en el registro RD.

EJEMPLO

AND R5, R7, R9 : R5 = R7&R9

AND R2, R2, R4 : R2 = R2&R4

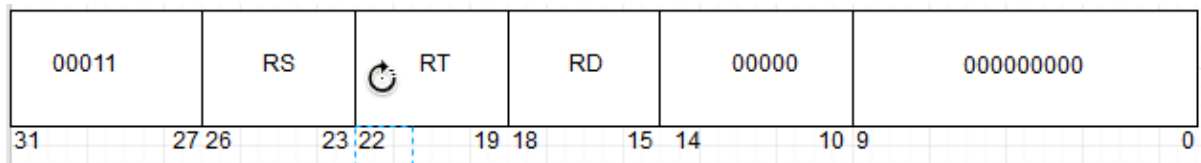
OR

Or lógico

FORMATO ENSAMBLADOR

OR RD, RS, RT

CODIFICACIÓN



OPERACIÓN

if (bit[31:27] == 00011)

RD = RS | RT

DESCRIPCIÓN

En caso de que el código de operación indique un or, aplica este operador lógico entre los registros RS y RT y almacena el resultado en el registro RD.

EJEMPLO

OR R5, R7, R9 : R5 = R7|R9

OR R2, R2, R4 : R2 = R2|R4

NOR

Nor lógico

FORMATO ENSAMBLADOR

NOR RD, RS, RT

CODIFICACIÓN

00100	RS	RT	RD	00000	000000000
31	27 26	23 22	19 18	15 14	10 9 0

OPERACIÓN

if (bit[31:27] == 00100)

RD = RS NOR RT

DESCRIPCIÓN

En caso de que el código de operación indique un nor, aplica este operador lógico entre los registros RS y RT y almacena el resultado en el registro RD.

EJEMPLO

NOR R5, R7, R9 : R5 = R7 NOR R9

NOR R2, R2, R4 : R2 = R2 NOR R4

SLL Desplazamiento lógico a la izquierda

FORMATO ENSAMBLADOR

SLL RD, RT, #SHAMT

CODIFICACIÓN

00101	0000	RT	RD	SHAMT	000000000
31	27 26	23 22	19 18	15 14	10 9 0

OPERACIÓN

if (bit[31:27] == 00101)

RD = RT << SHAMT

DESCRIPCIÓN

En caso de que el código de operación indique un SLL, aplica un desplazamiento a la izquierda del número guardado en el registro RT la cantidad de veces que lo indique el valor del *shift amount* (SHAMT) y guarda el resultado en el registro RD.

EJEMPLO

SLL R5, R7, #2 : R5 = R7 << 2

SLL R2, R2, #4 : R2 = R2 << 4

NOTA

Nótese que el registro RS no se utiliza en esta operación.

SRL Desplazamiento lógico a la derecha

FORMATO ENSAMBLADOR

SRL RD, RT, #SHAMT

CODIFICACIÓN

00110	0000	RT	RD	SHAMT	000000000
31	27 26	23 22	19 18	15 14	10 9 0

OPERACIÓN

if (bit[31:27] == 00110)

RD = RT >> SHAMT

DESCRIPCIÓN

En caso de que el código de operación indique un SRL, aplica un desplazamiento a la derecha del número guardado en el registro RT la cantidad de veces que lo indique el valor del *shift amount* (SHAMT) y guarda el resultado en el registro RD.

EJEMPLO

SLL R5, R7, #2 : R5 = R7 >> 2

SLL R2, R2, #4 : R2 = R2 >> 4

NOTA

Nótese que el registro RS no se utiliza en esta operación.

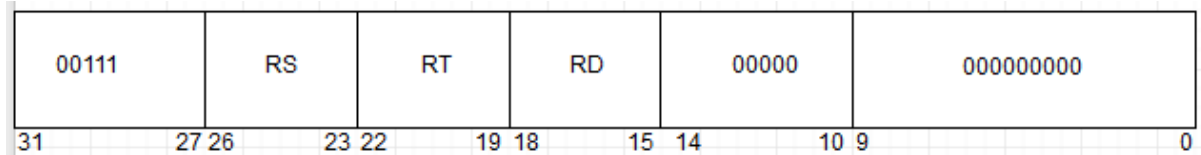
MULT

Multiplicación

FORMATO ENSAMBLADOR

MULT RD, RS, RT

CODIFICACIÓN



OPERACIÓN

if (bit[31:27] == 00111)

RD = RS * RT

DESCRIPCIÓN

En caso de que el código de operación de la instrucción sea el establecido para la multiplicación, se almacena el resultado de dicha operación entre los registros RS y RT en el registro RD.

EJEMPLO

MULT R5, R7, R9 : R5 = R7*R9

MULT R2, R2, R4 : R2 = R2*R4

NOTA

Se utiliza el símbolo ‘*’ para denotar la multiplicación entre dos registros.

DIV

División

FORMATO ENSAMBLADOR

DIV RD, RS, RT

CODIFICACIÓN

01000	RS	RT	RD	00000	000000000
31	27 26	23 22	19 18	15 14	10 9 0

OPERACIÓN

if (bit[31:27] == 01000)

RD = RS / RT

DESCRIPCIÓN

En caso de que el código de operación de la instrucción sea el establecido para la división, divide el valor del registro RS entre el valor del registro RT y almacena el resultado en el registro RD.

EJEMPLO

DIV R5, R7, R9 : R5 = R7/R9

DIV R2, R2, R4 : R2 = R2/R4

NOTA

Se utiliza el símbolo '/' para denotar la división entre dos registros.

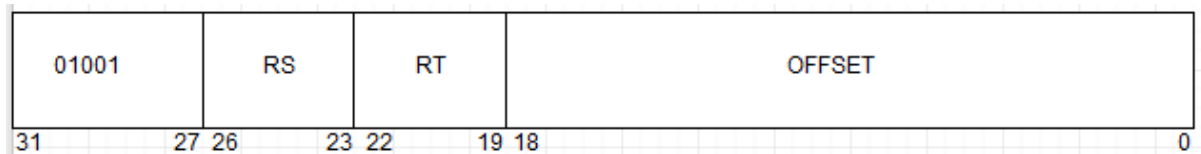
BEQ

Branch on equal

FORMATO ENSAMBLADOR

BEQ RS, RT, offset

CODIFICACIÓN



OPERACIÓN

```
if (bit[31:27] == 01001)
    if (RS == RT)
        PC = offset[9:0]
    else
        PC = PC+1
```

DESCRIPCIÓN

Si el código de operación de la instrucción es el establecido para el BEQ, compara los registros RS y RT. En caso de que los registros contengan el mismo valor, el PC toma el valor inmediato (offset) de la instrucción y ejecuta la instrucción que allí se encuentra.

EJEMPLO

BEQ R2, R3, 100 : Si R2=R3, PC = 100
BEQ R5, R7, 64 : Si R5=R7, PC = 64

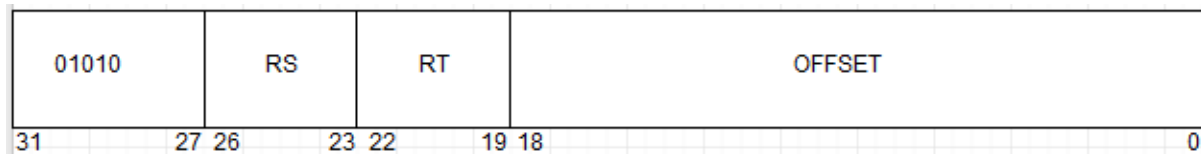
BNE

Branch not equal

FORMATO ENSAMBLADOR

BNE RS, RT, offset

CODIFICACIÓN



OPERACIÓN

if (bit[31:27] == 01010)

 if (RS != RT)

 PC = offset[9:0]

 else

 PC = PC+1

DESCRIPCIÓN

Si el código de operación de la instrucción es el establecido para el BEQ, compara los registros RS y RT. En caso de que los registros contengan valores diferentes, el PC toma el valor inmediato (offset) de la instrucción y ejecuta la instrucción que allí se encuentra.

EJEMPLO

BNE R2, R3, 100 : Si R2≠R3, PC = 100

BNE R5, R7, 64 : Si R5≠R7, PC = 64

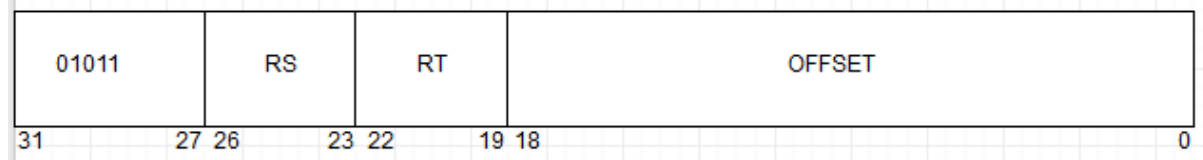
LW

Cargar palabra

FORMATO ENSAMBLADOR

LW RS, offset(RT)

CODIFICACIÓN



OPERACIÓN

if (bit[31:27] == 01011)

RS = MEM(RT + offset)

DESCRIPCIÓN

Si el código de operación de la instrucción es el establecido para el LW, toma el valor en memoria que se encuentra en la posición RT sumado con el número inmediato u *offset* y almacena el valor de la palabra encontrada en el registro RS.

EJEMPLO

LW R2, 100(R3) : R2 = MEM(R3 + 100)

LW R5, 256(R7) : R5 = MEM(R7 + 256)

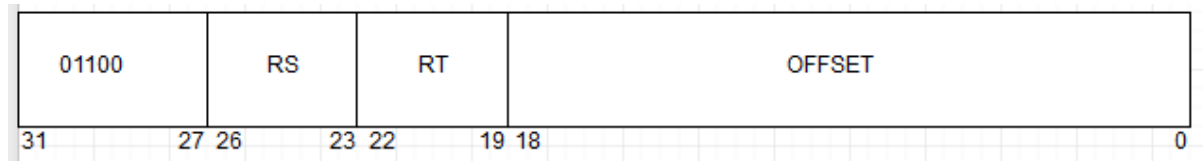
LB

Cargar byte

FORMATO ENSAMBLADOR

LB RS, offset(RT)

CODIFICACIÓN



OPERACIÓN

if (bit[31:27] == 01100)

RS = MEM(RT + offset[7:0])

DESCRIPCIÓN

Si el código de operación de la instrucción es el establecido para el LB, toma el valor en memoria que se encuentra en la posición RT sumado con el número inmediato u *offset* y almacena el valor del byte (8 bits) encontrado en el registro RS.

EJEMPLO

LB R2, 100(R3) : R2 = MEM(R3 + 100_{7:0})

LB R5, 256(R7) : R5 = MEM(R7 + 256_{7:0})

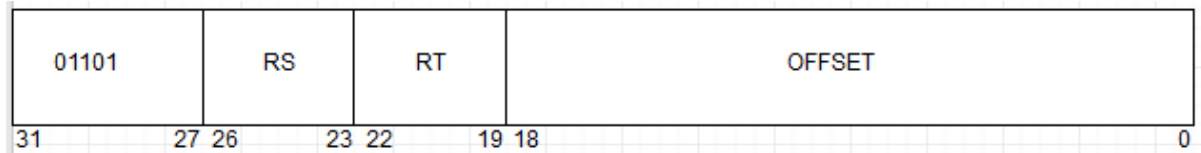
SW

Almacenar palabra

FORMATO ENSAMBLADOR

SW RS, offset(RT)

CODIFICACIÓN



OPERACIÓN

if (bit[31:27] == 01101)

MEM(RT + offset) = RS

DESCRIPCIÓN

Si el código de operación de la instrucción es el establecido para el SW, toma el valor del registro RS y lo almacena en la posición de memoria de RT + Inmediato (MEM[Rt+Inm] = Rs).

.

EJEMPLO

SW R2, 100(R3) : MEM(R3 + 100) = R2

SW R5, 256(R7) : MEM(R7 + 256) = R5

5.EXCEPCIONES

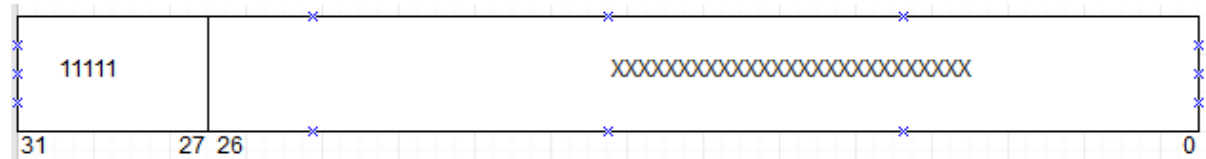
NOP

No realiza operación alguna

FORMATO ENSAMBLADOR

NOP

CODIFICACIÓN



OPERACIÓN

if (bit[31:27] == 11111)

PC = PC +1

DESCRIPCIÓN

Si el código de operación de la instrucción es el establecido para el NOP, esto quiere decir que en ese ciclo de reloj no va a realizarse ninguna operación con el fin de evitar riesgos en la ejecución de las instrucciones que componen el programa.

.

EJEMPLO

NOP : PC = PC+1

NOTA

El formato de la instrucción NOP es particular. Se estableció de esta forma con el fin de manejarlo de la manera más flexible posible con respecto al diseño propuesto de la microarquitectura.

6.Referencia rápida de JOF32

RD – Registro de destino

RS, RT – Registros de operandos

PC – Program Counter (Contador de programa)

MEM – Memoria

OFF – Offset (Desplazamiento)

CONST – Constante

ADDR – Sumador

OPERACIONES ARITMÉTICAS	
ADD RD, RS, RT	$RD = RS + RT$
SUB RD, RS, RT	$RD = RS - RT$
ADDI RD, RS, CONST19	$RD = RS + CONST19$
MULT RD, RS, RT	$RD = RS \times RT$
DIV RD, RS, RT	$RD = RS / RT$

OPERACIONES DE SHIFT	
SLL RS, RT, INMEDIATO	$RS = RT \ll INMEDIATO$
SRL RS, RT, INMEDIATO	$RS = RT \gg INMEDIATO$

OPERACIONES LÓGICAS	
AND RD, RS, RT	$RD = RS \& RT$
OR RD, RS, RT	$RD = RS RT$
NOR RD, RS, RT	$RD = \neg(RS RT)$
NOP	NO-OP

INSTRUCCIONES DE LOAD Y STORE	
SW RT, OFF19(RS)	$MEM(RS + OFF19) = RT$
LW RT, OFF19(RS)	$RT = MEM(RS + OFF19)$
LB RT, OFF19(RS)	$RT = MEM(RS + OFF19)_{7:0}$

OPERACIÓN DE SALTO Y BRANCHES	
J ADDR10	$PC = PC[31:27]::ADDR27$
BEQ RS, RT, OFF19	IF $RS = RT$, $PC = OFF10$
BNE RS, RT, OFF19	IF $RS \neq RT$, $PC = OFF10$