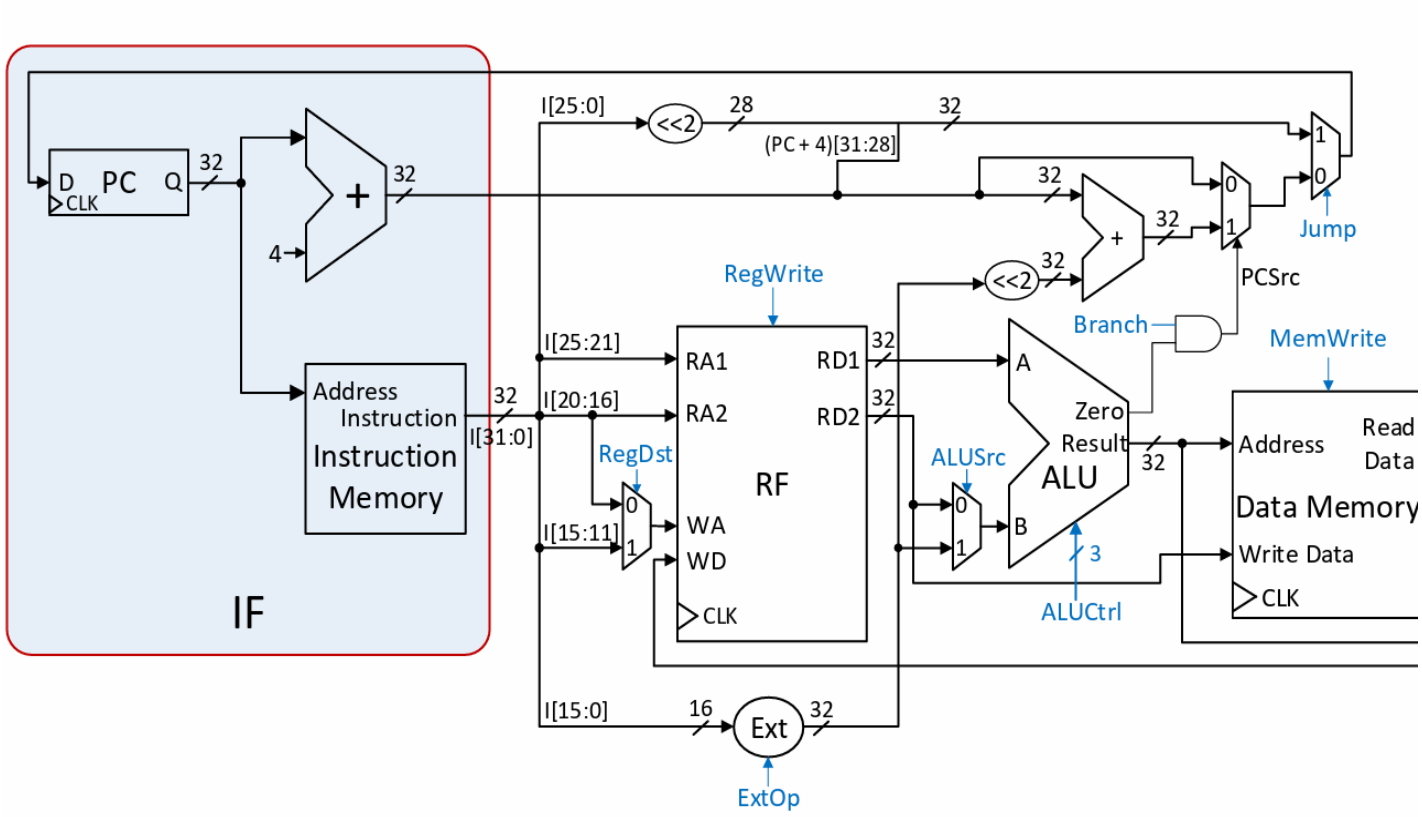
TIP		Sintaxa	RTL
Tip R			
Addition	**add**	add $d, $s, $t	$d ← $s + $t; PC ← PC + 4;
Subtraction	**sub**	sub $d, $s, $t	$d ← $s - $t; PC ← PC + 4;
Shift Left Logical (with shift amount – sa)	**sll**	sll $d, $t, h	$d ← $t << h; PC ← PC + 4;
Shift Right Logical (with shift amount – sa)	**srl**	srl $d, $t, h	$d ← $t >> h; PC ← PC + 4;
Logical AND	**and**	and $d, $s, $t	$d ← $s & $t; PC ← PC + 4;
Logical OR	**or**	or $d, $s, $t	$d ← $s | $t; PC ← PC + 4;
Logical XOR	**xor**	xor $d, $s, $t	$d ← $s ^ $t; PC ← PC + 4;
Shift-Right Arithmetic	**sra**	sra $d, $t, h	$d ← $t >> h; PC ← PC + 4;
Tip I			
Add Immediate	**addi**	addi $t, $s, imm	$t ← $s + SE(imm); PC ← PC + 4;
Load Word	**lw**	lw $t, offset($s)	$t ← MEM[$s + SE(offset)]; PC ← PC + 4;
Store Word	**sw**	sw $t, offset($s)	MEM[$s + SE(offset)] ← $t; PC ← PC + 4;
Branch on Equal	**beq**	beq $s, $t, offset	if $s == $t then PC ← PC + 4 + (SE(offset) << 2) else PC ← PC + 4;
Branch on Greater Than Zero	**bgtz**	bgtz $s, offset	If $s > 0 then PC ← PC + 4 + (SE(offset) << 2) else PC ← PC + 4;
Branch on Greater than or Equal to Zero	**bgez**	bgez $s, offset	if $s >= 0 then PC ← PC + 4 + (SE(offset) << 2) else PC ← PC + 4;
Tip J	**j**	j addr	PC ← (PC + 4)[31:28] || (addr << 2);

```
function CMMDC(a, b)
  if a = 0
    return b
  while b ≠ 0
    if a > b
      a := a − b
    else
      b := b − a
  return a
```

```
offset 00:    lw $1, offset($10) --a
offset 04:    addi $10, $10, 4
offset 08:    lw $2, offset($10) --b

offset 12:    addi $0, $0, 8 --adr mem

offset 16:    beq $1, $30, offset 72

offset 20:    beq $2, $30, offset 64

offset 24:    sw $1 offset($0)
offset 28:    addi $0, $0, 4
offset 32:    sw $2 offset($0)
offset 36:    addi $0, $0, 4

offset 40:    sub $3, $1, $2
offset 44:    bgtz $3, offset 56
offset 48:    sub $2, $2, $1
offset 52:    j addr 5
offset 56:    sub $1, $1, $2
offset 60:    j addr 5

offset 64:    sw $1 offset($0)
offset 68:    j addr 19
offset 72:    sw $2 offset($0)
offset 76:
```

```
100011_01010_00001_
001000_01010_01010_
100011_01010_00010_

001000_00000_00000_

000100_00001_11110_

000100_00010_11110_

101011_00000_00001_
001000_00000_00000_
101011_00000_00010_
001000_00000_00000_

000000_00001_00010_
000111_00011_00000_
000000_00010_00001_
000010_00000000000
000000_00001_00010_
000010_00000000000

101011_00000_00001_
000010_00000000000
101011_00000_00010_
```


IF
D
PC
Q
CLK
32
4
+
32
Address
Instruction
Instruction
Memory
32
I[31:0]
I[25:0]
<<2
28
32
(PC + 4)[31:28]
1
0
Jump
32
+
32
0
1
32
<<2
PCSrc
RegWrite
Branch
MemWrite
I[25:21]
I[20:16]
RegDst
I[15:11]
RA1
RA2
WA
WD
CLK
RF
RD1
RD2
32
32
A
B
ALU
Zero
Result
32
ALUSrc
3
ALUCtrl
Address
Read
Data
Data Memory
Write Data
CLK
I[15:0]
16
Ext
32
ExtOp

FORMAT

000000 sssss ttttt ddddd 00000 100000
000000 sssss ttttt ddddd 00000 100010
000000 sssss ttttt ddddd hhhhh 000000
000000 00000 ttttt ddddd hhhhh 000010
000000 sssss ttttt ddddd 00000 100100
000000 sssss ttttt ddddd 00000 100101
000000 sssss ttttt ddddd 00000 100110
000000 00000 ttttt ddddd hhhhh 000011

001000 sssss ttttt iiiiiiiiiiiiiii
100011 sssss ttttt oooooooooooooooo
101011 sssss ttttt oooooooooooooooo
000100 sssss ttttt oooooooooooooooo
000111 sssss 00000 oooooooooooooooo
000001 sssss 00000 oooooooooooooooo
000010 aaaaaaaaaaaaaaaaaaaaaaaa

_0000000000000000	8D410000
_0000000000000100	214A0004
_0000000000000000	8D420000
_0000000000001000	20000008
_0000000001001000	103E0048
_0000000001000000	105E0040
_0000000000000000	56008000
_0000000000000100	20000004
_0000000000000000	AC020000
_0000000000000100	20000004
_00011_00000_100010	00221822
_0000000000111000	1C600038
_00010_00000_100010	00411022
00000000000101	04000005
_00001_00000_100010	00220822
00000000000101	04000005
_0000000000000000	AC010000
00000000010011	04000013
_0000000000000000	AC020000