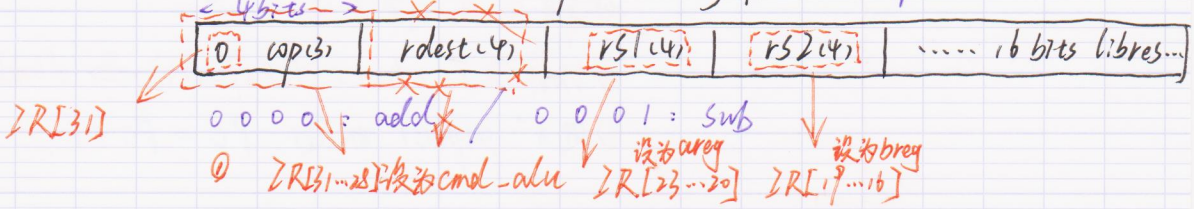
Architecture des Ordinateurs

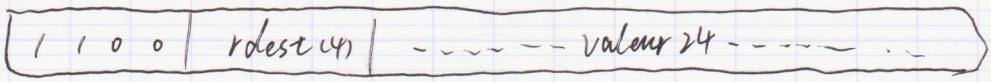
TD1

• 1. Les instructions.

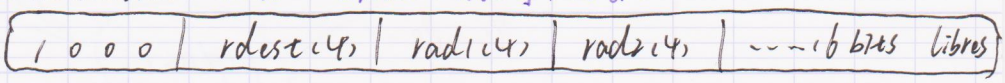
(1) Des instructions arithmétiques et logiques : op rs1, rs2, rdest



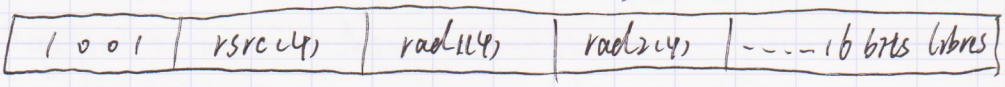
(2) L'instruction Set valeur24, rdest



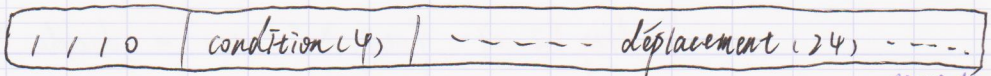
(3) L'instruction Load [rad1 + rad2], rdest



(4) L'instruction Store rsrc, [rad1 + rad2]



(5) Les instruction branchement branchement (b_condition_adresse)



déplacement24 = adresse de branchement − adresse courante

Go

r14 = Le registre PC (Program Counter) : 保当前指令的地址

r15 = Le registre IR (Instruction Register) : 指令寄存器，当前指令的代码.

Store = PC ← 第一条指令的地址

重复

IR ← [PC] // 将当前指令代码存入IR

执行该指令

转到下一条指令 : PC ← PC+1 / PC ← adresse de branchement

直到程序结束

• La forme d'un circuit séquentiel 时序电路形式.

areg = 1110, abus → r14(PC)

dbus|n : 10

dreg : 1111 dbus → r15(IR)

1. 起始状态 : "fetch"

⇒ 2. 通过"IR ← [PC]"操作，我们从状态"fetch"转为"decode"

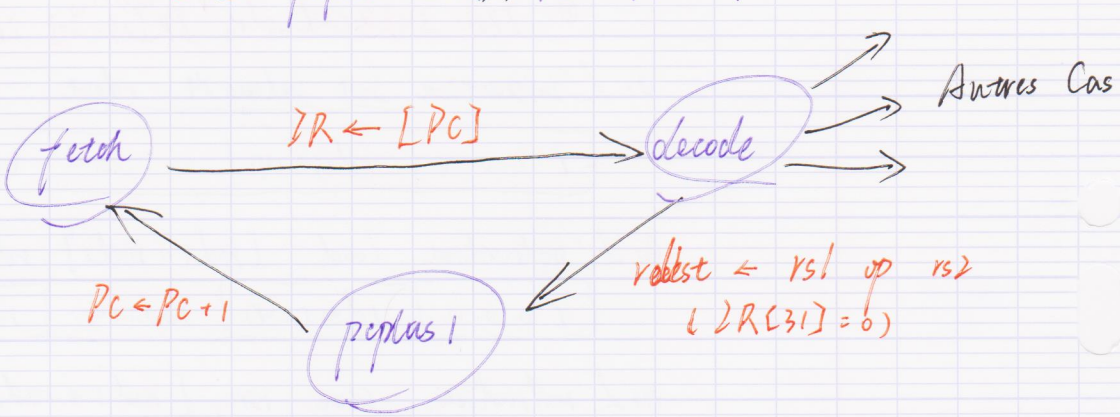
3. "decode" 状态开始，分为以下几种可能性：

a) Instructions arithmétiques et logiques (即 IR[31]=0)

① 译从第1页阅笔记)

② 结果保存 rdest (IR[27...24]) ⇒ dbusIn = 01 ⇒ dreg (IR[27...24])

③ 进入 "pcplus1" 状态，PC ← PC+1.



areg[3...0] = fetch * "1110" + setflags * IR[23...20] + pcplus1 * "1110"

breg[3...0] = fetch * "0000" + decode * IR[19...16] + pcplus1 * "0001"

dreg[3...0] = fetch * "1111" + decode * IR[27...24] + pcplus1 * "1110"

wrdcmd[3...0] = fetch * "0000" + decode * IR[31...28] + pcplus1 * "0000"

dbusIn[1...0] = fetch * "01" + decode * "01" + pcplus1 * "01"

write = fetch * "0" + decode * "0" + pcplus1 * "0"

Architecture des Ordinateurs

Instructions.

* reg : 寄存器
* cst : 常数
* adr : 地址

add	%ri, reg/cst, %rj	加法	%rj ← %ri + reg/cst
addcc	%ri, reg/cst, %rj	加(进位)	
sub		减法	%rj ← %ri - reg/cst
subcc		减(进位)	
umulcc		乘(进位)	%rj ← %ri × reg/cst
udivcc		除(进位)	%rj ← %ri ÷ reg/cst
andcc		AND	
orcc		OR	
xorcc	(同为0, 异为1)	异或	
xnorcc	(同为1, 异为0)	同或	
sll		左移	%rj ← %ri << reg/cst
srl		右移	%rj ← %ri >> reg/cst
sethi	val22, %ri	SET	%ri ← val22 (末位为22 bits)
ld	[%ri + reg/cst], %rj	Load	32 bits
ldub			8 bits
st	%ri, [%rj + reg/cst]	STORE	32 bits
stb			8 bits
call	adr	调用	
ba	adr	无条件跳转	
bcond	adr	如果条件成立(cond)跳转	
be			Branch Equal (Z)
bne			Branch Not Equal (not Z)
bneg			Branch NEGative (N)
bpos			Branch POSitive (not N)
bcs			Branch on Carry Set (C)
bcc			Branch on Carry Clear (not C)
bvs			Branch on oVerflow Set (V)
bvc			Branch on oVerflow Clear (not V)
bg	not (Z or (N xor V))	大于时转	Branch on Greater (not (Z or (N xor V))
bge	not (N xor V)	大于等于时转	Branch on Greater or Equal
bl	N xor V	小于时转	Branch on Less
ble	Z or (N xor V)	小于等于时转	Branch on Less or Equal.

Branchement →

Synthétiques →

bgu	not (Z or C)		Branch on Greater, Unsigned.
bcc	not C		Branch on Greater or equal, unsigned.
bcs	C		Branch on less than, unsigned
bleu	Z or C		Branch on Less or Equal, Unsigned.
clr %ri		CLEAR	orcc %r0, %r0, %ri
mov %ri, %rj		%rj ← %ri	orcc %ri, %r0, %rj
inccc %ri		自加1	addcc %ri, 1, %ri
notcc %ri, %rj		%rj ← %ri的补码	xnorcc %ri, %ri, %rj
deccc %ri		自减1	subcc %ri, 1, %ri
set val31...0, %ri		%ri ← val	sethi val31...10, %ri ; orcc %ri, val9...0, %ri
setq val12...0, %ri			orcc %ri, Val12...0, %ri
cmp %ri, %rj		比较	subcc %ri, %rj, %r0
tst %ri		符号及溢出意义检测	orcc %ri, %r0, %r0
negcc %ri		0 - %ri	subcc %r0, %ri, %ri
nop		无操作	sethi 0, %r0
jmp %ri		跳转至绝对地址	
push %ri		入栈	subcc %r30, 4, %r30 ; st %ri, [%r30]
pop %ri		出栈	ld [%r30], %ri ; addcc %r30, 4, %r30