Architecture des Ordinateurs

TD1

• 1. Les instructions.

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

< 4 bits >

0 cop(3)	rdest(4)	rs1(4)	rs2(4)	...... 16 bits libres...

IR[31]

0 0 0 0 : add / 0 0 0 1 : sub

① IR[31...28]设为cmd_alu   设为areg IR[23...20]   设为breg IR[19...16]

(2) L'instruction Set valeur24, rdest

1 1 0 0	rdest(4)	------ valeur24 ------

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

1 0 0 0	rdest(4)	rad1(4)	rad2(4)	---16 bits libres

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

1 0 0 1	rsrc(4)	rad1(4)	rad2(4)	----16 bits libres

(5) Les instruction branchement   branchement (b_condition_adresse)

1 1 1 0	condition(4)	------ déplacement(24) -----

déplacement24 = adresse de branchement − adresse ~~courant~~ courante

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.

```
Start :  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.

1. 起始状态 : "fetch"

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

areg : 1110,
abus → r14(PC)

dbusIn : 10

dreg : 1111
dbus → r15(IR)

3. "decode" 状态开始，分为以下几种可能性：

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

①详见第1页同色笔记

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

③进入"pcplus1"状态，PC ← PC + 1.

$$\begin{tikzcd}
\text{fetch} \arrow[rr, "IR \leftarrow [PC]"] & & \text{decode} \arrow[r] \arrow[r, shift left=2] \arrow[r, shift right=2] \arrow[dl, "\substack{rdest \leftarrow rs1\ op\ rs2 \\ (IR[31]=0)}"] & \text{Autres Cas} \\
& \text{pcplus1} \arrow[ul, "PC \leftarrow PC+1"] & &
\end{tikzcd}$$

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"

ualcmd[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 (无符号22bits)
ld	[%ri + reg/cst], %rj	load	32bits
ldub			8 bits
st	%ri, [%rj + reg/cst]	STORE	32bits
stb			8 bits
call	adr	调用	

Branchement →

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.

Synthétiques →

bgu	not (Z or C)	Branch on Greater, Unsigned.
bcc	not C	Branch on Greater or Equal, ~~unsi~~ 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 $val_{31...0}$, %ri	%ri ← val	sethi $val_{31...10}$, %ri ; orcc %ri, $val_{9...0}$, %ri
setq $val_{12...0}$, %ri		orcc %ri, $Val_{12...0}$, %ri
cmp %ri, %rj	比较	subcc %ri, %rj, %r0
tst %ri	符号及有无意义检测	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	sub~~cc~~ %r30, 4, %r30; st %ri, [%r30]
pop %ri	出栈	ld [%r30], %ri ; add %r30, 4, %r30