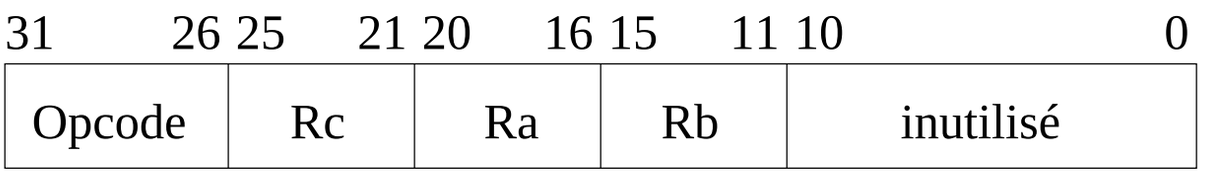
Une réalisation de la machine β orientée vers les performances

Machine β : format des instructions (rappel)

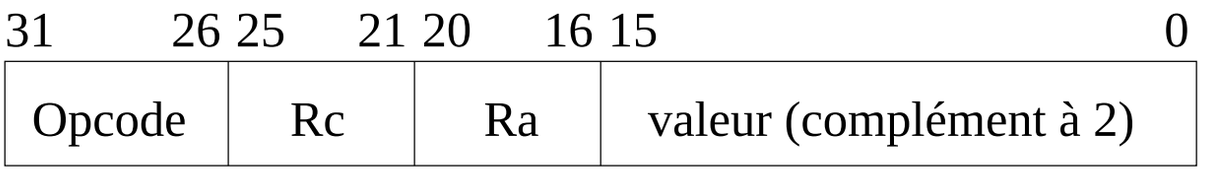
Toutes les opérations de la machine β (sauf les accès à la mémoire) sont effectuées entre registres.

Il y a deux formats possibles pour les instructions :

- Le format sans valeur constante (literal)



- Le format avec valeur constante (literal)



Machine β : instructions arithmétiques (rappel)

Sans valeur		Avec valeur	
Opcode	nom	Opcode	nom
$0x20$	ADD	$0x30$	ADDC
$0x21$	SUB	$0x31$	SUBC
$0x22$	MUL	$0x32$	MULC
$0x23$	DIV	$0x33$	DIVC

ADD(Ra,Rb,Rc) : $PC \leftarrow PC + 4$
$Reg[Rc] \leftarrow Reg[Ra] + Reg[Rb]$

ADDC(Ra,literal,Rc) : $PC \leftarrow PC + 4$
$Reg[Rc] \leftarrow Reg[Ra] + SEXT(literal)$

SEXT(literal) représente la valeur contenue dans l'instruction convertie de 16 à 32 bits, le signe étant conservé.

Pour mémoire, les instructions de comparaison, les instructions logiques et les instructions de décalage ont une structure similaire.

Machine β : instructions d'accès à la mémoire (rappel)

Opcode	nom
$0x18$	LD
$0x19$	ST

LD(Ra,literal,Rc) : PC $\leftarrow$ PC + 4
EA $\leftarrow$ Reg[Ra] + SEXT(literal)
Reg[Rc] $\leftarrow$ Mem[EA]

ST(Rc,literal,Ra) : PC $\leftarrow$ PC + 4
EA $\leftarrow$ Reg[Ra] + SEXT(literal)
Mem[EA] $\leftarrow$ Reg[Rc]

EA est ce qu'on appelle l'adresse "effective".

Machine β : instructions d'accès à la mémoire II (rappel)

Opcode	nom
$0x1F$	LDR

LDR(label,Rc) : $PC \leftarrow PC + 4$
$EA \leftarrow PC + 4 \times SEXT(literal)$
$Reg[Rc] \leftarrow Mem[EA]$

Dans LDR, l'instruction fournie à l'assembleur contient un étiquette (label). Dans l'instruction assemblée, la valeur introduite (literal) est calculée par

$$literal = ((OFFSET(label) - OFFSET(inst.\ courante)) \div 4) - 1$$

Machine β : instructions de branchement (rappel)

Opcode	nom
$0x1B$	JMP

JMP(Ra,Rc) : PC $\leftarrow$ PC + 4
EA $\leftarrow$ Reg[Ra] & $0x$FFFFFFFC
Reg[Rc] $\leftarrow$ PC
PC $\leftarrow$ EA

Les 2 bits inférieurs de Ra sont mis à 0 pour être sûr d'avoir une adresse de mot ; l'adresse de l'instruction suivant le JMP est sauvée dans Rc.

Machine β : instructions de branchement II (rappel)

Opcode	nom
$0x1D$	BEQ/BF
$0x1E$	BNE/BT

BEQ(Ra,label,Rc) : $PC \leftarrow PC + 4$
$EA \leftarrow PC + 4 \times SEXT(literal)$
$TMP \leftarrow Reg[Ra]$
$Reg[Rc] \leftarrow PC$
if $TMP = 0$ **then** $PC \leftarrow EA$

Dans BEQ, l'instruction fournie à l'assembleur contient un étiquette (label). Dans l'instruction assemblée, la valeur introduite (literal) est calculée par

$$literal = ((OFFSET(label) - OFFSET(inst.\ courante)) \div 4) - 1$$

BNE est similaire à BEQ si ce n'est que l'on teste la nonégalité à 0.

TMP est utilisé car Ra et Rc pourraient être identiques.

Une réalisation de la machine β sans micro-code

Le but est de permettre l'exécution d'une instruction en un cycle. Dans ce but on procède comme suit.

- On multiplie les chemins de transfert de données en vue de permettre des transferts simultanés.
- La plupart des éléments ont alors un multiplexeur en entrée permettant de sélectionner une source parmi plusieurs.
- On utilise un ensemble de registres *à trois ports* (*three port*) permettant simultanément deux lectures et une écriture (à la transition d'horloge suivante).
- La logique de contrôle a alors pour but de sélectionner les entrées des différents éléments et la fonction calculée par l'ALU.
- On sépare la mémoire d'instructions et la mémoire de données (caches).

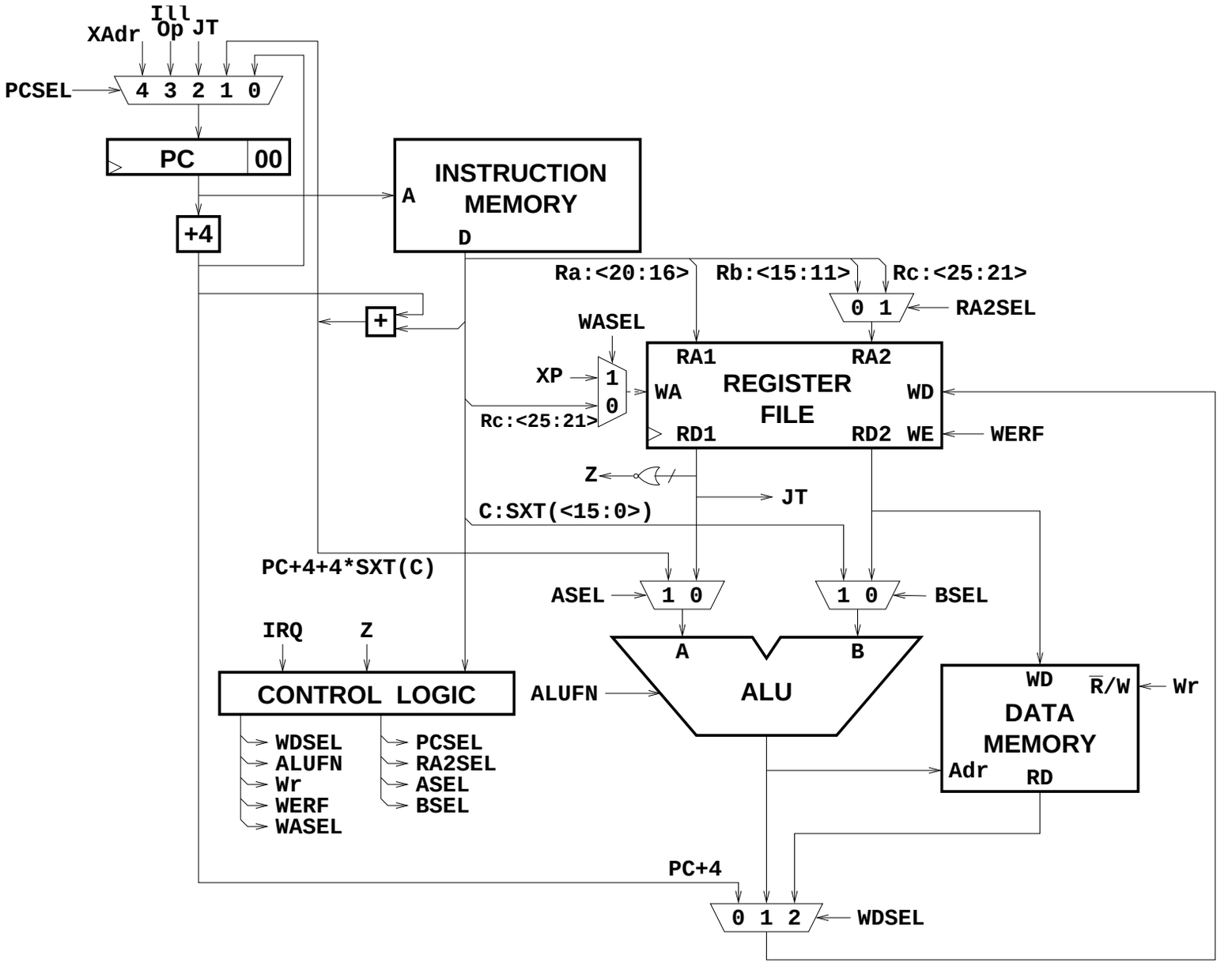
Une réalisation de la machine β sans micro-code (suite)

- L'écran suivant présente un shéma général d'une réalisation de β sans micro-code.

- Les "exceptions" qui forcent le sauvetage de `PC + 4` dans `XP` (registre 30) et le saut à une adresse prédéterminée sont prévues.

- Le mode superviseur n'est pas explicitement implémenté.

- La mémoire virtuelle n'est pas explicitement prévue (on peut supposer une cache travaillant en adresses virtuelles).


Ill
XAdr
Op
JT
PCSEL
4 3 2 1 0
PC
00
+4
INSTRUCTION MEMORY
A
D
+
Ra:<20:16>
Rb:<15:11>
Rc:<25:21>
0 1
RA2SEL
WASEL
XP
1
0
Rc:<25:21>
RA1
RA2
WA
REGISTER FILE
WD
RD1
RD2
WE
WERF
Z
JT
C:SXT(<15:0>)
PC+4+4*SXT(C)
ASEL
1 0
1 0
BSEL
IRQ
Z
A
B
ALU
ALUFN
CONTROL LOGIC
WDSEL
ALUFN
Wr
WERF
WASEL
PCSEL
RA2SEL
ASEL
BSEL
WD
R/W
Wr
DATA MEMORY
Adr
RD
PC+4
0 1 2
WDSEL

Machine β sans micro-code : les signaux

`Xadr`	adresse du handler d'interruption.
`ILLOP`	adresse du handler d'opération illégale
`JT`	adresse de destination d'un `JMP` (jump target)
`PCSEL`	sélection de la source du PC
`RA2SEL`	sélection de la 2ième adresse de lecture pour les registres
`WASEL`	sélection de l'adresse d'écriture dans les registres
`XP`	registre d'exception (30)
`WERF`	Write Enable Register File
`Z`	test à 0 de la sortie 1 des registres
`A(B)SEL`	sélection entrée A (B) de l'ALU
`WDSEL`	sélection des données à écrire dans les registres
`ALUFN`	sélection de la fonction de l'ALU
`Wr`	mémoire de données en mode "écriture"

Machine β sans micro-code : la logique de contrôle

Cette logique est implémentée par une ROM ou un PLA.

	OP	OPC	LD	ST	JMP	BEQ	BNE	LDR	ILLOP	trap
ALUFN	F(op)	F(op)	"+"	"+"	-	-	-	"A"	-	-
WERF	1	1	1	0	1	1	1	1	1	1
BSEL	0	1	1	1	-	-	-	-	-	-
WDSEL	1	1	2	-	0	0	0	2	0	0
Wr	0	0	0	1	0	0	0	0	0	0
RA2SEL	0	-	-	1	-	-	-	-	-	-
PCSEL	0	0	0	0	2	Z==1	Z==0	0	3	4
ASEL	0	0	0	0	-	-	-	1	-	-
WASEL	0	0	0	-	0	0	0	0	1	1

Machine β sans micro-code : performances ?

- La machine présentée exécute une instruction par cycle, mais les cycles devront être très long.

- En effet, il faut attendre, en cascade, la stabilisation de la mémoire d'instructions, des registres, de l'ALU, de la mémoire de données.

- La solution est d'organiser la machine en *pipeline*