IFT1225
ARCHITECTURE DES ORDINATEURS 2

Dép. d'IRO
Université de Montréal
(professeur E. Cerny)

Plan

Semaine **Sujets**

1: Révision de circuits logiques, organisation de base d'un ordinateur, jeu d'instructions
2: Langage machine, langage d'assemblage, l'assembleur de SRC
3: Mesures de performance, quelques processeurs réels (SPARC, MC68000, PPC601, Pentium)
4, 5: Accélération du traitement d'instructions: pipelines, émission d'instructions multiples; machines à format d'instruction large (VLIW), introduction aux multiprocesseurs.

Plan

Semaine **Sujets**

6 Unité de contrôle microprogrammée

7-9: Mémoire centrale: organisation, les composantes micro-électroniques (SRAM, DRAM, ROM, EPROM, EEPROM, FlashEPROM), modularisation, hiérarchie de mémoire: cache, mémoire virtuelle, mémoire secondaire; exemples.

10-12: Entrée / Sortie: organisation de base; le bus: synchrone / asynchrone; formes de transfert: programmée, par interruption, accès direct à la mémoire, processeurs E/S.

13: Introduction à l'Internet.

Évaluation

- Travaux pratiques 30% (4 à 5)
- Examen intra 30% (le 20 octobre)
- Examen final 40% (entre les 7 et 21 décembre)

Texte

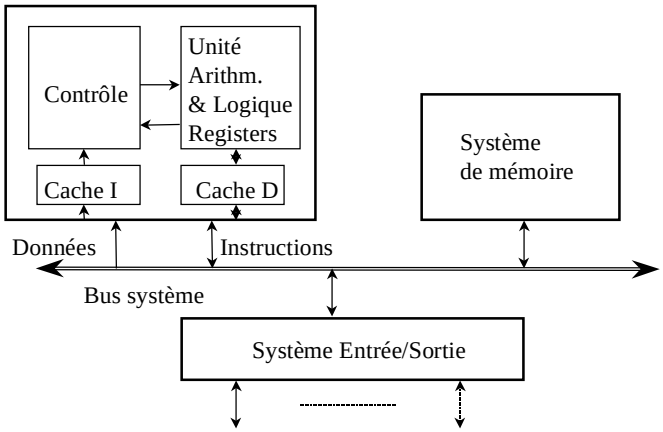
- Vincent P. Heuring, Harry F. Jordan: Computer Systems Design and Architecture; Addison Wesley Longman, Inc., 1997.

Révision IFT1213

- Architecture du jeu d'instruction (ISA)
- Organisation de base de l'unité centrale
- Portes et circuits logiques combinatoires
- Bascules bistables
- Machines à état fini (FSM), circuits séquentiels synchrones
- Unite arithmétique, transferts d'information sur un bus

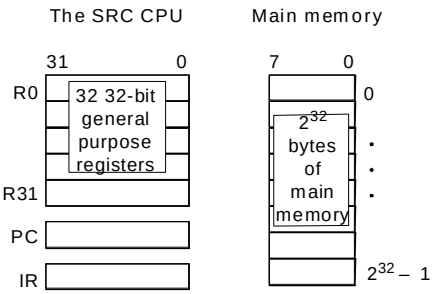
Organisation d'un ordinateur

Unité Centrale (CPU) typique



Ordinateur simple SRC

- Architecture RISC (*Reduced Instruction Set Computer*)
- Trois formats de base d'instructions
- 2 ou 3 spécifications d'opérandes/résultats dans l'instruction (dont un peut être dans la mémoire)



R[7] veut dire le contenu du registre 7

M[30] veut dire le contenu de la mémoire à l'adresse 30

État du processeur

PC⟨31..0⟩: **compteur d'instructions (adresse de mémoire de l'instruction suivante)**

IR⟨31..0⟩: **registre d'instructions**

Run: **indicateur *run/halt* (un bit)**

Strt: **signal start**

R[0..31]⟨31..0⟩: **registres généraux**

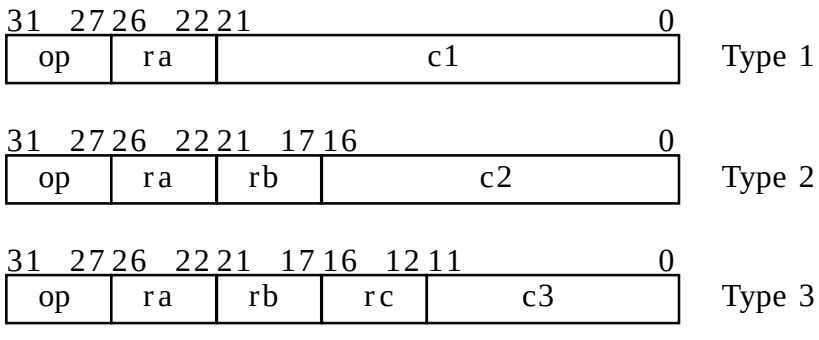
Format d'instructions

- Information fournie par une instruction:
  - Opération, Opérandes, Destination du résultat
  - Prochaine instruction
- Types d'opérations:
  - Chargement/Rangement (*Load/Store*)
    - ld r2, 4(r6) charger r2 par le mot de la mémoire à r[6]+4
    - st r2, 8(r6) ranger le contenu de r2 à la mémoire à r[6]+8
  - Aritmétique/Logique
    - add r1, r2, r3 ranger r[2]+r[3] dans r[1]
  - Branchement
    - brzr r2, r3 transfert de contrôle à l'adr. r[2] si r[3] = 0

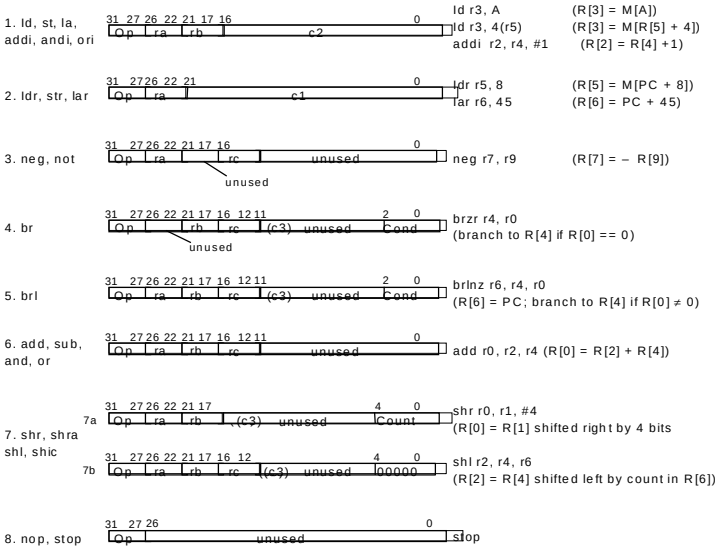
Formats d'instructions de SRC

Trois formats de base

Sept sousformats



Formats détaillés



Exemples

Chargement / Rangement

Instruction	op	ra	rb	c1	Meaning	Addressing Mode
ld r1, 32	1	1	0	32	R[1] ← M[32]	Direct
ld r22, 24(r4)	1	22	4	24	R[22] ← M[24+R[4]]	Displacement
st r4, 0(r9)	3	4	9	0	M[R[9]] ← R[4]	Register indirect
la r7, 32	5	7	0	32	R[7] ← 32	Immediate
ldr r12, -48	2	12	–	-48	R[12] ← M[PC -48]	Relative
lar r3, 0	6	3	–	0	R[3] ← PC	Register (!)

Exemples

Format	Example	Meaning
neg ra, rc	neg r1, r2	;Negate (r1 = -r2)
not ra, rc	not r2, r3	;Not (r2 = r3´)
add ra, rb, rc	add r2, r3, r4	;2's complement addition
sub ra, rb, rc		;2's complement subtraction
and ra, rb, rc		;Logical and
or ra, rb, rc		;Logical or
addi ra, rb, c2	addi r1, r3, #1	;Immediate 2's complement add
		;no subtract… c2 can be negative
andi ra, rb, c2		;Immediate logical and
ori ra, rb, c2		;Immediate logical or

Exemples

Deux formes de base d'instructions de branchement:

```
br rb, rc, c3<2..0>          ; aller à R[rb] si R[rc] satisfait
                             ; la condition spécifiée
                             ; par c3<2..0>

brl ra, rb,  rc, c3<2..0>    ; R[ra] ← PC
                             ; et brancher comme br
```

c3<2..0>, les 3 bits inf. de c3 déterminent la condition de branchement:

lsbs	condition	Assy language form	Example
000	never	brlnv	brlnv r6
001	always	br, brl	br r5, brl r5
010	if rc = 0	brzr, brlzr	brzr r2, r4, r5
011	if rc ≠ 0	brnz, brlnz	
100	if rc >= 0	brpl, brlpl	
101	if rc < 0	brmi, brlmi	

Exemple de branchement

En langage C: goto Toto

En langage d'assemblage SRC:

```
        lar  r1,  Toto    ; charger l'adresse dans r1
        br   r1           ; effectuer le branchement
        • • •
Toto    • • •
```


Exécution d'instructions

instruction_interpretation := (
¬Run∧Strt → Run ← 1:
Run → (IR ← M[PC]: PC ← PC + 4; instruction_execution));
recherche ***exécution***

instruction_execution := (
ld (:= op= 1) → R[ra] ← M[disp]: load register
ldr (:= op= 2) → R[ra] ← M[rel]: load register relative
st (:= op= 3) → M[disp] ← R[ra]: store register
str (:= op= 4) → M[rel] ← R[ra]: store register relative
la (:= op= 5) → R[ra] ← disp: load displ. address
lar (:= op= 6) → R[ra] ← rel: load relative address
nop (:= op= 0) → : No operation
stop (:= op= 31) → Run ← 0: Stop instruction
); instruction_interpretation.

Interprétation des champs de bits

ld (:= op= 1) → R[ra] ← M[disp]:

disp⟨31..0⟩ := ((rb=0) → c2⟨16..0⟩ {sign extend}:
(rb≠0) → R[rb] + c2⟨16..0⟩ {sign extend, 2's comp.}):

ld (:= op= 1) → R[ra] ← M[
((rb=0) → c2⟨16..0⟩ {sign extend}:
(rb≠0) → R[rb] + c2⟨16..0⟩ {sign extend, 2's comp.}):
]:

- Exemple:
  - If IR = 00001 00101 00011 00000000000001011
  - then ld → R[5] ← M[R[3] + 11]:

Exécution des instruction de branchement

- Condition déterminée par 3 bits inférieurs (*lsbs*) de l'instruction
- Après, le registre *Link* (R[ra]) pointe à l'instruction suivante, c.à.d. à l'adresse de retour de la procédure

cond := (c3⟨2..0⟩=0 → 0:	never
c3⟨2..0⟩=1 → 1:	always
c3⟨2..0⟩=2 → R[rc]=0:	if register is zero
c3⟨2..0⟩=3 → R[rc]≠0:	if register is nonzero
c3⟨2..0⟩=4 → R[rc]⟨31⟩=0:	if positive or zero
c3⟨2..0⟩=5 → R[rc]⟨31⟩=1):	if negative
br (:= op= 8) → (cond → PC ← R[rb]):	conditional branch
brl (:= op= 9) → (R[ra] ← PC:	
cond → (PC ← R[rb])):	branch and link

Arithmétique et logique

add (:= op=12) → R[ra] ← R[rb] + R[rc]:
addi (:= op=13) → R[ra] ← R[rb] + c2⟨16..0⟩
{2's comp. sign ext.}:
sub (:= op=14) → R[ra] ← R[rb] - R[rc]:
neg (:= op=15) → R[ra] ← -R[rc]:

and (:= op=20) → R[ra] ← R[rb] ∧ R[rc]:
andi (:= op=21) → R[ra] ← R[rb] ∧ c2⟨16..0⟩ {sign extend}:
or (:= op=22) → R[ra] ← R[rb] ∨ R[rc]:
ori (:= op=23) → R[ra] ← R[rb] ∨ c2⟨16..0⟩ {sign extend}:
not (:= op=24) → R[ra] ← ¬R[rc]:

2's complement sign extend = extension à un mot de 32 bits du bit de signe d'un nombre à complément de 2

Instructions de décalage

- **Compte n peut être 5 bits inf. d'un reg. ou de l'instruction**
- **Notation: @ - replication, # - concaténation**

n := ((c3⟨4..0⟩=0) → R[rc]⟨4..0⟩:
(c3⟨4..0⟩≠0) → c3⟨4..0⟩):

shr (:= op=26) → R[ra]⟨31..0⟩ ← (n @ 0) # R[rb]⟨31..n⟩:

shra (:= op=27) → R[ra]⟨31..0⟩ ← (n @ R[rb]⟨31⟩) # R[rb]⟨31..n⟩:

shl (:= op=28) → R[ra]⟨31..0⟩ ← R[rb]⟨31-n..0⟩ # (n @ 0):

shc (:= op=29) → R[ra]⟨31..0⟩ ← R[rb]⟨31-n..0⟩ # R[rb]⟨31..32-n⟩:

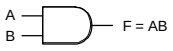
Formes typiques des modes d'adressage (pas tous dans SRC)

Registre cible

Nom	Assembleur	Transfert de regs	Utilisation
Register	Ra	R[t] ← R[a]	Var. temporaire
Register indirect	(Ra)	R[t] ← M[R[a]]	Pointeur
Immediate	#X	R[t] ← X	Constante
Direct, absolute	X	R[t] ← M[X]	Var. globale
Indirect	(X)	R[t] ← M[M[X]]	Pointeur
Indexed, based, or displacement	X(Ra)	R[t] ← M[X + R[a]]	Tableaux, structures
Relative	X(PC)	R[t] ← M[X + PC]	Vals dans pgm
Autoincrement	(Ra)+	R[t] ← M[R[a]]; R[a] ← R[a] + 1	Accès
Autodecrement	- (Ra)	R[a] ← R[a] - 1; R[t] ← M[R[a]]	séquentiel

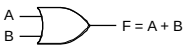
Le matériel ... Portes logiques

A	B	F
0	0	0
0	1	0
1	0	0
1	1	1



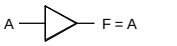
ET (AND)

A	B	F
0	0	0
0	1	1
1	0	1
1	1	1



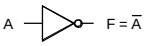
OU (OR)

A	F
0	0
1	1



TAMPON (BUFFER)

A	F
0	1
1	0



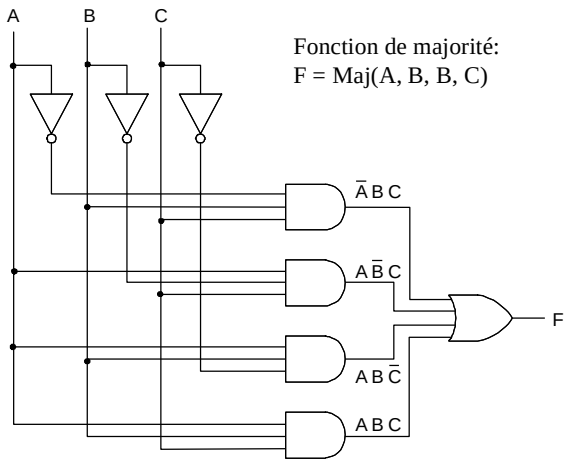
NON (NOT)

Autres:

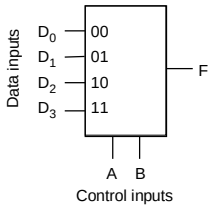
NON-ET (NAND)
NON-OU (NOR)
OU-EXCLUSIF (XOR)
NON-OU-EXCLUSIF (XNOR)
MULTIPLEXEUR (MUX)
DEMULTIPLEXEUR (DEMUX)

etc.

Circuit combinatoire

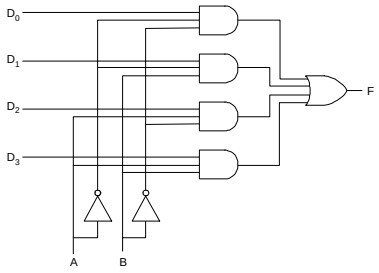


Multiplexeur

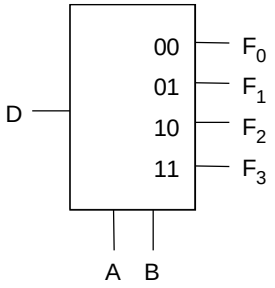


A	B	F
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

$$F = \bar{A}\,\bar{B}\,D_0 + \bar{A}\,B\,D_1 + A\,\bar{B}\,D_2 + A\,B\,D_3$$



Démultiplexeur



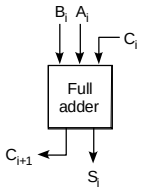
D	A	B	F_0	F_1	F_2	F_3
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

$F_0 = D \bar{A} \bar{B}$ $\quad F_2 = D A \bar{B}$

$F_1 = D \bar{A} B$ $\quad F_3 = D A B$

Comment construire un décodeur 2:4?

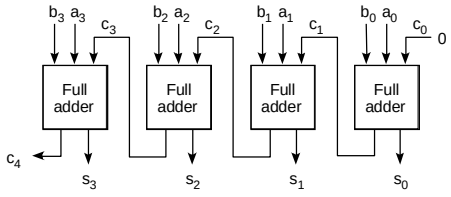
Additionneur



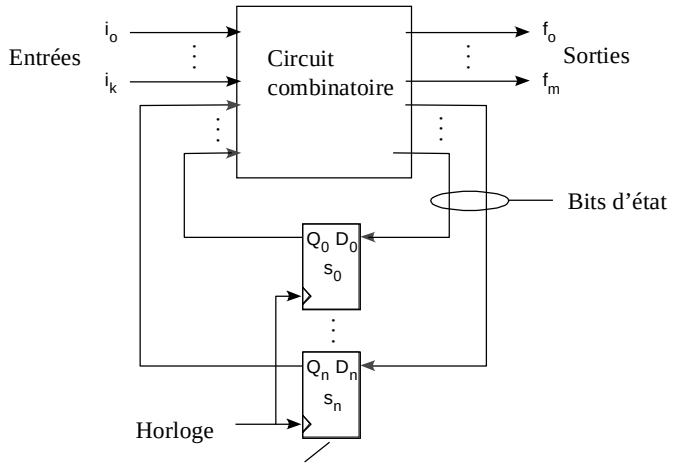
A_i	B_i	C_i	S_i	C_{i+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Carry In →	0	0	0	0	1	1	1	1
Operand A →	0	0	1	1	0	0	1	1
Operand B →	+ 0	+ 1	+ 0	+ 1	+ 0	+ 1	+ 0	+ 1
	0 0	0 1	0 1	1 0	0 1	1 0	1 0	1 1
	↑	↑						
	Carry Out	Sum						

Additionneur de 4 bits:

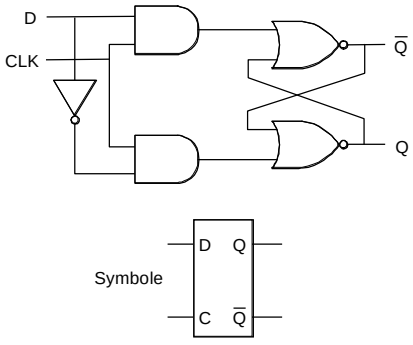


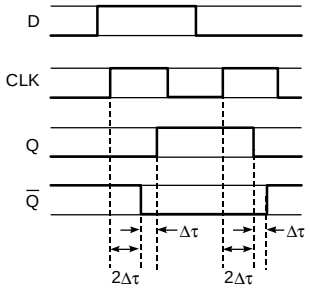
Circuits séquentiels



Retard (mémoire) - état du circuits
Machine à état fini (FSM)

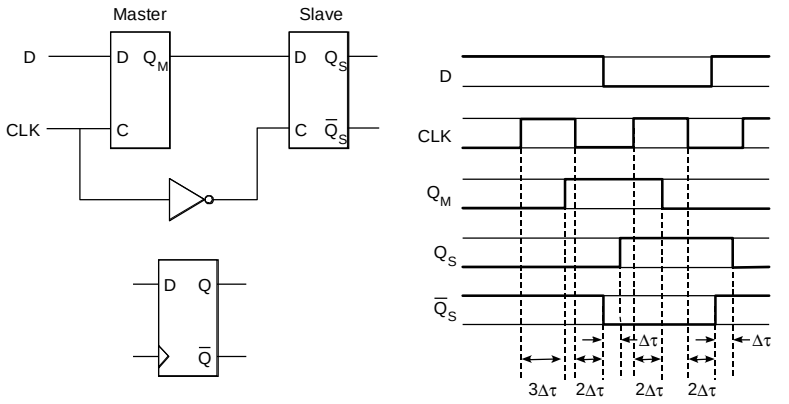
Bascule D synchrone



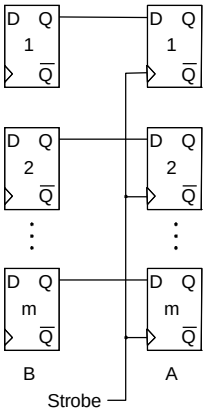


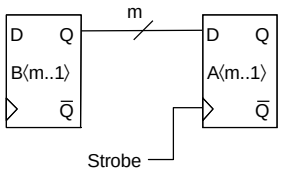
Délai d'une porte **Δτ**

Bascule maître-esclave (M/E)

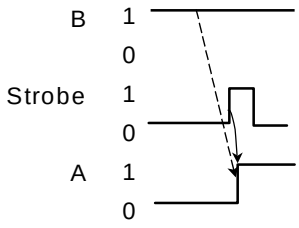


Transfert de données entre registres

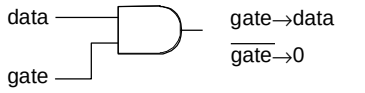




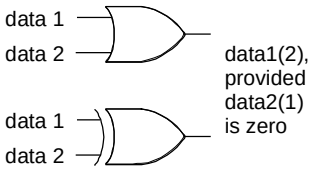
Synchronisation pour un bit



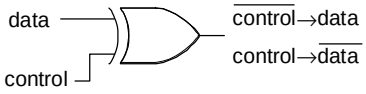
Commande de transmission de données avec des portes logiques



commande de données

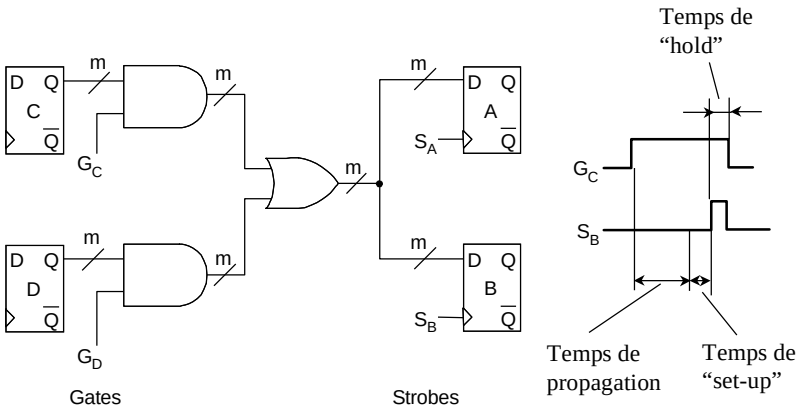


fusion de données



complément contrôlé

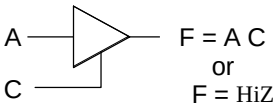
Transferts multiplexés entre plusieurs registres



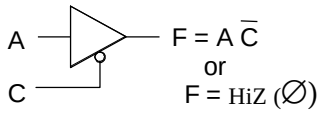
Bascules (M/E) sensibles au front montant de l'horloge !

Circuits tampons à trois états

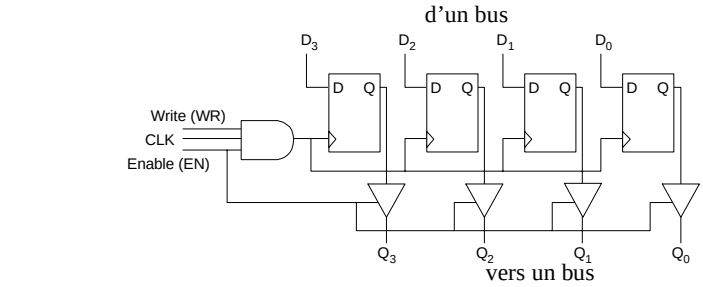
C	A	F
0	0	HiZ
0	1	HiZ
1	0	0
1	1	1

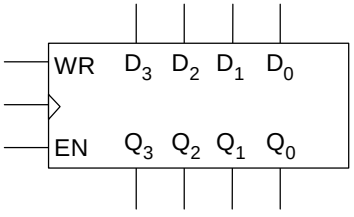


C	A	F
0	0	0
0	1	1
1	0	HiZ
1	1	HiZ



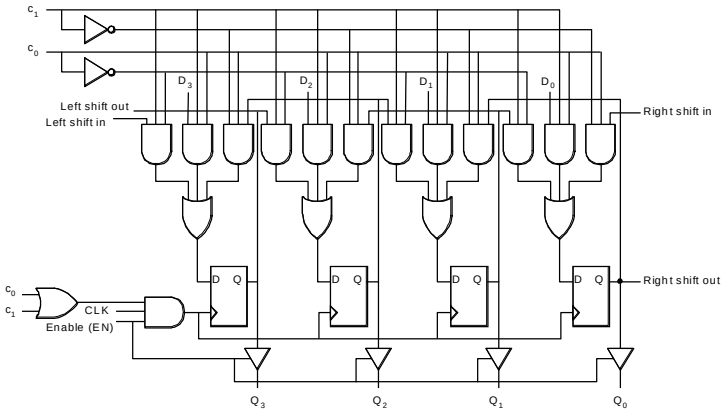
Registre à 8 bits



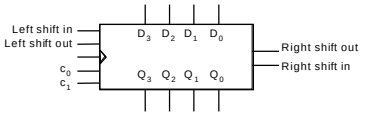


Comment ajouter à l'entrée un circuit à décalage à gauche/droit ...?

Registre avec un décaleur g/d à son entrée



Control c_1	Control c_0	Function
0	0	No change
0	1	Shift left
1	0	Shift right
1	1	Parallel load



Registres et unités arithmétiques et logiques liés par un bus

Exemple:

Spec. RT abstraite

R[3] ← R[1]+R[2];

RT concrète (implantation)

Y ← R[2];
Z ← R[1]+Y;
R[3] ← Z;

Séquence de contrôle

$R[2]_{out}$, Y_{in};
$R[1]_{out}$, Z_{in};
Z_{out}, $R[3]_{in}$;