Cours d'architecture

IUT Informatique S1

Patrick Magnaud

(d'après Daniel Dours & Roland Facca)

L'ordinateur

Système informatique

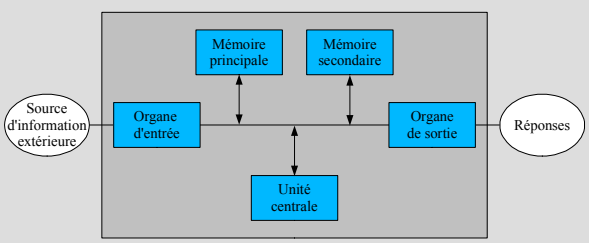
- Traitement automatique d'une information pré enregistrée
  - Information
    - Concept abstrait à coder sous forme symbolique
  - Traitement de l'information
    - Symboles codés transformés en d'autres symboles codés
  - Automatisation
    - Algorithme

L'ordinateur

- Composantes
  - Traitements
    - Circuits électroniques en tout ou rien
    - Actions de base : additionner, comparer, mémoriser, ...
  - Mémoire
    - Recevoir, conserver, restituer
    - Mémoire principale (très rapide)
    - Mémoire secondaire (très grosse capacité)
  - Organes d'accès
    - Communication avec l'extérieur
    - Entrée : clavier, souris, lecteurs, ...
    - Sortie : écran, imprimante, ...

L'ordinateur

- Fonctionnement



L'ordinateur

- Langages
  - Langage interne
    - Langage machine lié au matériel
  - Langages externes
    - Langages d'application
      - de haut niveau
      - d'assemblage
    - Langages de commandes
      - Exécution de programmes
      - Manipulation de fichiers
- Transformation d'un langage externe en langage interne
  - Interprétation
  - Traduction : compilation ou assemblage

Architecture en couche des ordinateurs

- Conception descendante
  - Similaire aux affinages en algorithmique
- Machine virtuelle
  - Machine physique + Logiciel
- Ordinateur constitué de 3 couches
  - Couche langage externe
    - Vue par le programmeur d'application
  - Couche machine
    - Vue par le programmeur système
  - Couche physique
    - Vue par le concepteur de machine
- Entouré de 2 autres couches

Couche langage externe

- Programmes utilisateurs traduits à ce niveau
  - Compilation ou assemblage
  - Édition de liens
- Problème de portabilité : 3 solutions
  - Langage de programmation universel
    - Exemple : Ada
  - Couche machine unique
    - Exemple : Compatible PC
  - Machine virtuelle
    - Exemple : Java Virtual Machine (JVM)

Couche machine

- Décomposée en 3 niveaux
  - Niveau système
    - Lien avec la couche langage externe
  - Niveau macro machine (architecture)
    - Programmée en langage d'assemblage
    - Une machine simple comporte 3 composants
      - Le processeur central (CPU)
      - La mémoire principale (ou centrale)
      - Des dispositifs d'entrée/sortie (E/S)
    - Interconnectés par des bus
  - Niveau micro machine (organisation)
    - Décomposition du processeur en
      - Partie(s) opérative(s)
      - Partie contrôle

Processeur central (CPU)

- Aussi appelé microprocesseur
- Constitué de 2 parties
  - Mémoire registre
    - Très petite
      - quelques dizaines d'octets
    - Très rapide
      - vitesse du processeur d'instructions
  - Processeur d'instructions
    - Rôle
      - exécuter le programme stocké en mémoire principale

Processeur d'instructions

- Prélever l'instruction courante
- L'exécuter
  - Interpréter
  - Éventuellement calculer l'adresse(s) mémoire et aller chercher le(s) opérande(s) en mémoire
  - Faire le calcul
  - Éventuellement ranger le résultat en mémoire
- Passer à l'instruction suivante à exécuter
  - Soit la suivante en mémoire (en séquence)
  - Soit l'instruction spécifiée dans l'instruction courante (branchement)

Mémoire principale

- Constituée de cellules ou mots mémoire
- Un mot est constitué d'octet(s)
- Capacité de quelques kibioctets (Kio) à quelques gibioctets (Gio)
- Temps d'accès uniforme entre 1 et 250 ns
- 2 catégories
  - Mémoire vive “RAM” (volatile)
    - Statique, dynamique
  - Mémoire morte ROM (non volatile)
    - ROM, PROM, EPROM, EEPROM
    - NOVRAM

Mémoire secondaire

- Disquettes
  - 1440 Kio
- Disques magnétiques
  - 4 Go à 1 To, 3600 à 15000 t/mn
- Disques optiques
  - CD (Compact Disc)
    - 650 à 700 Mio (74 ou 80 mn)
    - 48 X à 56 X, 1 X = 150 ko/s
  - DVD (Digital Versatile Disc)
    - 4,7 à 17,1 Go
    - 16 X, 1 X = 1,35 Mo/s
- Autres
  - Mémoire flash

Niveau composants électroniques

- Base de la couche physique
- Le composant de base est le transistor utilisé comme interrupteur électronique
- 2 technologies
  - Technologie bipolaire (ancien)
    - TTL, ECL
  - Technologie MOS (actuel)
    - PMOS, NMOS, CMOS
- 2 caractéristiques importantes
  - Temps de commutation
  - Puissance dissipée

Niveau composants logiques

- Assemblage de transistors
- Basé sur l'algèbre de Boole
- Georges Boole, logicien anglais, travaillant sur le langage dans les années 1850
- Portes logiques
  - ET, OU, NON, NON-OU, NON-ET, ...
- Transforment une ou plusieurs entrées logiques (représentées par 0 ou 1) en une sortie logique

Niveau circuits logiques

- Assemblage de portes logiques
- 2 sortes de circuits
  - Combinatoire : les sorties ne dépendent que des entrées au “même” instant
    - Décodeur, multiplexeur, comparateur, additionneur, décaleur, ...
    - Unité arithmétique et logique (UAL)
  - Séquentiel : les sorties dépendent aussi des entrées antérieures
    - Registre, compteur, registre à décalage, mémoire, ...
- Réalisés dans des circuits intégrés
  - Plusieurs centaines de millions de transistors sur une puce

Représentations de l'information

Information

- Information manipulée par l'ordinateur
  - Programmes
  - Données
- Écrits par l'utilisateur dans un langage de codification
  - Langage externe
  - Alphabet alphanumérique
- Problème de traduction en langage binaire
  - Instructions
  - Données

Données

- 3 types d'information
  - Logiques
  - Caractères et chaînes de caractères
  - Nombres
    - Entiers naturels (non signés)
    - Entiers relatifs (signés)
    - Flottants
- Plusieurs formats
  - Bit, digit (4 bits), octet (8 bits)
  - Demi-mot, mot, double-mot

Langage

- Alphabet
  - $A = \{a_1, a_2, ..., a_p\}$ est un ensemble fini de symboles dont le cardinal $|A| = p$ est appelé base
- Mots
  - suites finies de symboles (concaténation)
  - Format des mots
    - Fixe
    - Variable
- Un langage est un sous-ensemble de A^* ensemble des suites finies de symboles de A

Construction d'un langage

- Morphologie des mots
  - Soit on construit tous les mots possibles
    - Structure morphologique non restrictive
  - Soit on ajoute des règles pour supprimer des mots
    - Structure morphologique restrictive
- Un langage peut être défini
  - En compréhension
    - Alphabet, format, éventuellement restrictions
  - En extension
    - Énumération de tous les mots
- On peut obtenir tous les mots en construisant un arbre

Puissance lexicographique

- Cardinal de l'ensemble des mots du langage
  - Soit
    - L un langage
    - M_L l'ensemble des mots de L
  - La puissance lexicographique de L, notée $|L|$, est donc définie comme le cardinal de M_L
    - $|L| = |M_L|$
  - Si L est à structure morphologique non restrictive et p est la base de l'alphabet de L
    - En format fixe $= n$, $|L| = p^n$
    - En format variable $\leq n$, $|L| = \frac{p^{n+1} - p}{p - 1}$

Ordre lexicographique

- Soit L un langage d'alphabet A dont les symboles sont ordonnés
- Soit X et Y deux mots de L tels que

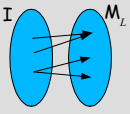
$$\begin{cases} X = x_1 x_2 \dots x_q \\ Y = y_1 y_2 \dots y_r \end{cases} \quad \text{avec } x_i, y_i \in A$$

- X et Y sont dans l'ordre lexicographique, noté $X \leq Y$, si et seulement si

$$(\forall i \leq n, x_i = y_i \text{ et } q \leq r)$$
$$ou \quad (\exists j \leq n, x_j < y_j \text{ et } \forall i < j, x_i = y_i)$$
$$avec \quad n = min(q, r)$$

Assignation sémantique

- Un langage sert à transmettre de l'information
- Il faut séparer
  - Les mots : forme, syntaxe
  - Le sens des mots : sémantique
- L'assignation sémantique définit le sens des mots du langage
- Dans un langage naturel
  - Un mot peut définir plusieurs “objets”
  - Un “objet” peut porter plusieurs noms
- Dans un langage informatique
  - Assignations multiples interdites



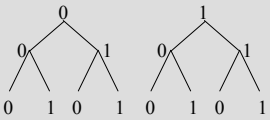
Codification

- Soit $\mathbf{I}$ un ensemble informationnel et L un langage
- La codification est une assignation sémantique qui est une application injective de $\mathbf{I}$ dans $\mathbf{M}_L$
- Une condition nécessaire est $|\mathbf{I}| \leq |L|$
- En général, s'il existe un ordre sur $\mathbf{I}$, la codification doit le respecter

Langage binaire

- Un langage binaire L est défini par
  - Un alphabet $\mathbf{A}$ = {0, 1} (base = 2)
  - Un format n
- S'il est à structure morphologique non restrictive
  - Format fixe : $|L| = 2^n$
  - Format variable : $|L| = 2^{n+1}-2$
- En général, les langages utilisés en machine sont de format fixe et à structure morphologique non restrictive

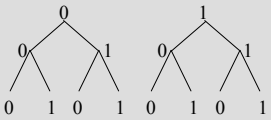
Exemple de langage binaire de format fixe 3



On obtient les mots de longueur 3 dans l'ordre lexicographique en parcourant les branches de l'arbre de la gauche vers la droite et en ne prenant que les mots de longueur 3

$\mathbf{M}_L$ = {000, 001, 010, 011, 100, 101, 110, 111}

Exemple de langage binaire de format variable ≤ 3



On obtient les mots de longueur 3 dans l'ordre lexicographique en parcourant les branches de l'arbre de la gauche vers la droite et en prenant tous les mots

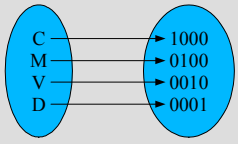
$\mathbf{M}_L$ = {0, 00, 000, 001, 01, 010, 011, 1, 10, 100, 101, 11, 110, 111}

Représentation des éléments informationnels

- La représentation machine s'effectue à l'aide d'éléments physiques à deux états
- D'où l'utilisation d'un langage binaire
- On peut utiliser 2 techniques
  - Représentation positionnée
    - appelée aussi primitive ou naturelle
  - Représentation codifiée
    - appelée aussi codée ou compacte

Représentation positionnée

- Système de représentation où chaque élément informationnel est déterminé par une position prise dans un ensemble
- Exemple
  - Supposons qu'on veuille représenter la situation de famille d'un individu qui peut être
    - Célibataire (C)
    - Marié (M)
    - Divorcé (D)
    - Veuf (V)



Représentation codifiée

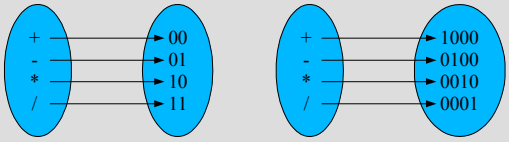
- Chaque élément est représenté par un mot d'un langage binaire de format fixe
- Il faut donc vérifier la contrainte $|\mathtt{I}| \leq |L|$, soit pour un langage binaire $|\mathtt{I}| \leq 2^n$
- On veut de plus le format le plus petit d'où la contrainte $|\mathtt{I}| > 2^{n-1}$
- On obtient donc $2^{n-1} < |\mathtt{I}| \leq 2^n$

Utilisation des 2 représentations

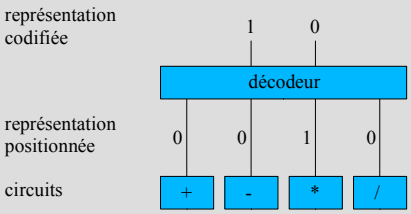
- La représentation positionnée est la seule utilisable pour la commande d'organes physiques
- La représentation codifiée est plus compacte, elle est donc utilisée quand il faut stocker ou transmettre de l'information
- Pour commander un organe physique, il faudra donc transformer une représentation codifiée en représentation positionnée
  - On utilise pour ça un décodeur

Exemple d'un organe de calcul

- Exemple
  - Soit un organe de calcul qui peut effectuer les 4 opérations arithmétiques +, -, *, /
  - Supposons qu'il existe un circuit distinct pour chaque opération
  - On doit passer d'une représentation codifiée à une représentation positionnée

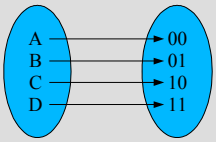


Utilisation d'un décodeur



Représentation binaire d'un langage externe

- Il suffit de
  - définir une codification binaire des symboles de l'alphabet du langage externe
  - remplacer chaque symbole par son code binaire
- Exemple
  - Soit un langage L dont l'alphabet est {A, B, C, D}
  - Et le mot de L égal à BAC
  - Sa représentation dans la codification binaire définie ci-dessous est 010010



Codage de l'alphabet

- Dans les langages informatiques, l'alphabet utilisé est composé de
  - lettres,
  - chiffres,
  - ponctuation,
  - symboles mathématiques,
  - etc...
- Le codage doit respecter l'ordre des lettres et des chiffres
- On utilise un langage dont le format varie de 6 à 8 bits, soit de 64 à 256 mots

Code ISO 8859

- Le code le plus couramment utilisé dans le monde occidental est le code ISO 8859
  - Il utilise 8 bits (256 mots)
  - C'est une extension du code ASCII
    - Code ASCII sur 7 bits (128 mots)
  - Les 128 premiers mots correspondent à ceux du code ASCII avec un 0 en tête
  - Il existe plusieurs variantes
    - Ce sont les 128 derniers mots qui changent
    - La variante utilisée en europe de l'ouest est le code ISO 8859-1, appelé aussi Latin-1
    - Une variante plus récente, ISO 8859-15, appelée aussi Latin-9, intègre, entre autres, le symbole €

Code ISO 8859-1 (latin-1)

	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0000	NUL	DLE		0	@	P	`	p			DCS		°	À	Ð	à	ð
0001	SOH	DC1	!	1	A	Q	a	q			PU1	¡	±	Á	Ñ	á	ñ
0010	STX	DC2	"	2	B	R	b	r	BPH		PU2	¢	²	Â	Ò	â	ò
0011	ETX	DC3	#	3	C	S	c	s	NBH		STS	£	³	Ã	Ó	ã	ó
0100	EOT	DC4	$	4	D	T	d	t			CCH	¤	´	Ä	Ô	ä	ô
0101	ENQ	NAK	%	5	E	U	e	u	NEL		MW	¥	µ	Å	Õ	å	õ
0110	ACK	SYN	&	6	F	V	f	v	SSA		SPA	¦	¶	Æ	Ö	æ	ö
0111	BEL	ETB	'	7	G	W	g	w	ESA		EPA	§	·	Ç	×	ç	÷
1000	BS	CAN	(	8	H	X	h	x	HTS		SOS	¨	¸	È	Ø	è	ø
1001	HT	EM	)	9	I	Y	i	y	HTJ			©	¹	É	Ù	é	ù
1010	LF	SUB	*	:	J	Z	j	z	VTS		SCI	ª	º	Ê	Ú	ê	ú
1011	VT	ESC	+	;	K	[	k	{	PLD		CSI	«	»	Ë	Û	ë	û
1100	FF	FS	,	<	L	\	l	|	PLU		ST	¬	¼	Ì	Ü	ì	ü
1101	CR	GS	-	=	M	]	m	}	RI		OSC	-	½	Í	Ý	í	ý
1110	SO	RS	.	>	N	^	n	~	SS2		PM	®	¾	Î	Þ	î	þ
1111	SI	US	/	?	O	_	o	DEL	SS3		APC	¯	¿	Ï	ß	ï	ÿ

Code ISO 8859-15 (latin-9)

	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0000	NUL	DLE		0	@	P	`	p			DCS		°	À	Ð	à	ð
0001	SOH	DC1	!	1	A	Q	a	q			PU1	¡	±	Á	Ñ	á	ñ
0010	STX	DC2	"	2	B	R	b	r	BPH		PU2	¢	²	Â	Ò	â	ò
0011	ETX	DC3	#	3	C	S	c	s	NBH		STS	£	³	Ã	Ó	ã	ó
0100	EOT	DC4	$	4	D	T	d	t			CCH	€	Ž	Ä	Ô	ä	ô
0101	ENQ	NAK	%	5	E	U	e	u	NEL		MW	¥	µ	Å	Õ	å	õ
0110	ACK	SYN	&	6	F	V	f	v	SSA		SPA	Š	¶	Æ	Ö	æ	ö
0111	BEL	ETB	'	7	G	W	g	w	ESA		EPA	§	·	Ç	×	ç	÷
1000	BS	CAN	(	8	H	X	h	x	HTS		SOS	š	ž	È	Ø	è	ø
1001	HT	EM	)	9	I	Y	i	y	HTJ			©	¹	É	Ù	é	ù
1010	LF	SUB	*	:	J	Z	j	z	VTS		SCI	ª	º	Ê	Ú	ê	ú
1011	VT	ESC	+	;	K	[	k	{	PLD		CSI	«	»	Ë	Û	ë	û
1100	FF	FS	,	<	L	\	l	|	PLU		ST	¬	Œ	Ì	Ü	ì	ü
1101	CR	GS	-	=	M	]	m	}	RI		OSC	-	œ	Í	Ý	í	ý
1110	SO	RS	.	>	N	^	n	~	SS2		PM	®	Ÿ	Î	Þ	î	þ
1111	SI	US	/	?	O	_	o	DEL	SS3		APC	¯	¿	Ï	ß	ï	ÿ

Unicode

- Permet de représenter les caractères dans la plupart des langues du monde
  - Les caractères sont représentés sur 32 bits, bien que actuellement les codes définis en utilisent au plus 21
  - Cette norme défini les mêmes codes que la norme ISO 10646
  - Les codes sont notés U+xxxx où xxxx est le code représenté en base 16, sur 4 à 6 chiffres
  - Les 256 premiers caractères sont ceux de la norme ISO 8859-1

UTF-8

- C'est une représentation d'Unicode qui utilise de 1 à 4 octet(s) pour un caractère

Représentation binaire UTF-8	Signification
0xxxxxxx	1 octet codant 1 à 7 bits
110xxxxx **10**xxxxxx	2 octets codant 8 à 11 bits
1110xxxx **10**xxxxxx **10**xxxxxx	3 octets codant 12 à 16 bits
11110xxx **10**xxxxxx **10**xxxxxx **10**xxxxxx	4 octets codant 17 à 21 bits

- Exemples

Caractère	Numéro du caractère	Codage binaire UTF-8
A	$65 = 1000001_2$	**0**1000001
é	$233 = 11101001_2$	**110**00011 **10**101001
€	$8364 = 10000010101100_2$	**1110**0010 **10**000010 **10**101100

Contrôle de l'information

- Lors d'un transfert d'information entre un émetteur et un récepteur, il peut se produire des erreurs
- On peut introduire une codification qui permet de
  - Détecter les erreurs (code auto vérificateur)
  - Corriger les erreurs (code auto correcteur)
- Pour cela, on doit utiliser un nombre de bits ***supérieur*** à celui strictement nécessaire
  - On parle de code redondant
- Dans ce qui suit, on considérera un maximum de 2 erreurs par message envoyé

Contrôle de parité simple

- Permet de détecter une seule erreur
- Si l'information est codée sur *n* bits, on en rajoute 1 de telle sorte que la somme des bits soit :
  - Paire (code à parité paire)
  - Impaire (code à parité impaire)
- Exemple
  - Soit à transmettre la chaîne 512 en ASCII en utilisant la parité paire

00110101 10110001 10110010
5 1 2

Contrôle de parité bi-dimensionnel

- Permet de corriger une erreur et d'en détecter deux
- L'information est transmise par paquet de plusieurs codes, avec un contrôle de parité sur chaque ligne et sur chaque colonne
- Exemple
  - Soit à transmettre la chaîne 512 en ASCII en utilisant la parité paire

00110101	5
10110001	1
10110010	2
00110110	

Contrôle de Hamming

- Permet de corriger une erreur sur chaque code transmit séparément
- Pour identifier la position de l'erreur, il faut introduire *k* bits de contrôle pour *m* bits d'information
- Le code comprends donc *m*+*k* bits qui peuvent être erronés
- Les *k* bits doivent pouvoir indiquer
  - soit qu'il n'y a pas d'erreur
  - soit la position de l'erreur, entre 1 et *m*+*k*

Contrôle de Hamming

- Il y a donc $m+k+1$ situations à distinguer ce qui entraîne la condition $m+k+1 \leq 2^k$
- On veut le plus petit nombre de bits de contrôle d'où $m+(k-1)+1 > 2^{k-1}$
- On obtient donc $2^{k-1} + 1 < m+k+1 \leq 2^k$
- Dans les exemples de la suite, on va considérer 4 bits de message et 3 bits de contrôle
  - Si $k = 2$ alors $0 < m \leq 1$
  - Si $k = 3$ alors $1 < m \leq 4$
  - Si $k = 4$ alors $4 < m \leq 11$

Contrôle de Hamming

- Bits de contrôle
  - Les bits de contrôle, notés k_1, k_2 et k_3, sont situés sur les positions puissances de deux : 1, 2 et 4
  - Les positions contrôlées par les bits k_1, k_2 et k_3 sont définies par les 1 dans le tableau suivant

position	k_3	k_2	k_1
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

Contrôle de Hamming

- Exemple de message envoyé
  - Soit le message de 4 bits 0101 à envoyer avec un contrôle de Hamming à parité paire

position	1	2	3	4	5	6	7
	k_1	k_2	m_1	k_3	m_2	m_3	m_4
message à coder			0		1	0	1
contrôle 1-3-5-7	0		0		1		1
contrôle 2-3-6-7		1	0			0	1
contrôle 4-5-6-7				0	1	0	1
message codé	0	1	0	0	1	0	1

Contrôle de Hamming

- Exemple de message reçu correct
  - Soit le message reçu suivant

position	1	2	3	4	5	6	7	
	k_1	k_2	m_1	k_3	m_2	m_3	m_4	
message reçu	0	1	0	0	1	0	1	
contrôle 1-3-5-7	0		0		1		1	correct (0)
contrôle 2-3-6-7		1	0			0	1	correct (0)
contrôle 4-5-6-7				0	1	0	1	correct (0)
message corrigé	0	1	0	0	1	0	1	

Contrôle = 000, le message reçu est correct

Contrôle de Hamming

- Exemple de message reçu avec 1 erreur
  - Soit le message reçu suivant

position	1	2	3	4	5	6	7	
	k_1	k_2	m_1	k_3	m_2	m_3	m_4	
message reçu	0	1	1	0	1	0	1	
contrôle 1-3-5-7	0		1		1		1	erroné (1)
contrôle 2-3-6-7		1	1			0	1	erroné (1)
contrôle 4-5-6-7				0	1	0	1	correct (0)
message corrigé	0	1	0	0	1	0	1	

Contrôle = 011, l'erreur est en position 3

Contrôle de Hamming

- Exemple de message reçu avec 2 erreurs
  - Soit le message reçu suivant

position	1	2	3	4	5	6	7	
	k_1	k_2	m_1	k_3	m_2	m_3	m_4	
message reçu	0	1	1	1	1	0	1	
contrôle 1-3-5-7	0		1		1		1	erroné (1)
contrôle 2-3-6-7		1	1			0	1	erroné (1)
contrôle 4-5-6-7				1	1	0	1	erroné (1)
message corrigé	0	1	1	1	1	0	0	

Contrôle = 111, l'erreur est en position 7,
ce qui introduit une 3ème erreur !!!

Contrôle de Hamming + parité

- Le contrôle de Hamming permet de distinguer
  - Pas d'erreur
  - 1 ou 2 erreurs
- Le contrôle de parité permet de distinguer
  - Pas d'erreur ou 2 erreurs
  - 1 erreur
- On peut donc combiner les deux pour améliorer le contrôle de Hamming
- Ce nouveau code permet de corriger une erreur et d'en détecter deux

Contrôle de Hamming + parité

- Les différentes situations possibles sont définies dans le tableau suivant

		Hamming	
		Correct (0)	Erroné (1 ou 2)
Parité	Correct (0 ou 2)	Correct	2 erreurs
	Erroné (1)	Parité erroné	1 erreur

Contrôle de Hamming + parité

- Exemple de message envoyé
  - Soit le message de 4 bits 0101 à envoyer avec un contrôle de Hamming à parité paire avec un bit supplémentaire de parité paire

position	0	1	2	3	4	5	6	7
	p	k_1	k_2	m_1	k_3	m_2	m_3	m_4
message à coder				0		1	0	1
contrôle 1-3-5-7		0		0		1		1
contrôle 2-3-6-7			1	0			0	1
contrôle 4-5-6-7					0	1	0	1
message codé	1	0	1	0	0	1	0	1

Contrôle de Hamming + parité

- Exemple de message reçu correct
  - Soit le message reçu suivant

position	0	1	2	3	4	5	6	7	
	p	k_1	k_2	m_1	k_3	m_2	m_3	m_4	
message reçu	1	0	1	0	0	1	0	1	
contrôle 1-3-5-7		0		0		1		1	correct
contrôle 2-3-6-7			1	0			0	1	correct
contrôle 4-5-6-7					0	1	0	1	correct
contrôle parité	1	0	1	0	0	1	0	1	correct
message corrigé	1	0	1	0	0	1	0	1	

Contrôle = 000, parité correcte, le message reçu est correct

Contrôle de Hamming + parité

- Exemple de message reçu avec 1 erreur
  - Soit le message reçu suivant

position	0	1	2	3	4	5	6	7	
	p	k_1	k_2	m_1	k_3	m_2	m_3	m_4	
message reçu	1	0	1	1	0	1	0	1	
contrôle 1-3-5-7		0		1		1		1	erroné (1)
contrôle 2-3-6-7			1	1			0	1	erroné (1)
contrôle 4-5-6-7					0	1	0	1	correct (0)
contrôle parité	1	0	1	1	0	1	0	1	erroné
message corrigé	1	0	1	0	0	1	0	1	

Contrôle = 011, parité fausse, l'erreur est en position 3

Contrôle de Hamming + parité

- Exemple de message reçu avec 2 erreurs
  - Soit le message reçu suivant

position	0	1	2	3	4	5	6	7	
	p	k_1	k_2	m_1	k_3	m_2	m_3	m_4	
message reçu	1	0	1	1	1	1	0	1	
contrôle 1-3-5-7		0		1		1		1	erroné (1)
contrôle 2-3-6-7			1	1			0	1	erroné (1)
contrôle 4-5-6-7					1	1	0	1	erroné (1)
contrôle parité	1	0	1	1	1	1	0	1	correct
message corrigé									

Contrôle = 111, parité correcte, il y a 2 erreurs, pas de correction