# Registres de la famille Intel 64

L'objectif de cette section est de connaître et savoir utiliser les registres de la famille Intel 64

## 1.1 Descriptif sommaire de la famille des processeurs Intel iAPx86

Il s'agit d'une famille de processeurs à compatibilité ascendante i.e. le code des plus anciens est toujours compris et correctement exécuté par les plus récents. Dans cette famille, on peut citer :

- 1978 le 8086 dispose d'un bus de 16 bits pour les données. Sa capacité d'adressage est de 1 Mo et il est cadencé à 4.77 ou 8 Mhz ;
- 1979 le 8088 est la version 8 bits du 8086. il a les mêmes caractéristiques, mais un assembleur plus réduit ;
- 1982 le 80286 est un microprocesseur 16 bits, pouvant adresser 8 Mo de code et de données, cadencé à 6,8,10,12 ou 16 Mhz. Il introduit deux modes d'adressages (réel ou protégé) ;
- 1985 le 80386 est un *faux* 32 bits, adressant 4 Go et cadencé à 16, 20 ou 25 Mhz ;
- 1989 le 80486 est un vrai 32 bits doté d'une mémoire cache intégrée et d'une unité de calcul en virgule flottante ;
- 1993 le 80586, appelé pentium pour des raisons de protection commerciale, dispose d'un bus de données de 64 bits et est muni d'un dispositif de prévision des branchements. Il est constitué de 2 processeurs en pipe-line parallèles lui permettant d'exécuter deux instructions en même temps. Initialement cadencé à 66 Mhz, son cadencement à 200 MHZ est mis sur le marché en 1996 ;
- 1995 le Pentium Pro est constitué de 6 millions de transistors (contre 275 000 pour le 386, 1 million pour le 486 et 3 pour le pentium) et la dimension des traits de ses transistors est de 0.6 microns (contre 11.5 pour le 386). Il peut traiter 250 millions d'instructions par seconde ;
- 2000 le Pentium III est cadencé à 1 Ghz.

En 2001, l'Itanium est lancé ; c'est un processeur disposant d'une architecture 64 bits. Les processeur X86 disposent maintenant d'une extension 64 bits à l'architecture IA32, mais sans rapport avec l'IA64 de l'itanium.

## 1.2 Registres généralistes de la famille des processeurs 34 bits Intel

Pour manipuler des données, les processeurs 64 bits de type Intel possèdent (parmi d'autres registres) quinze registres de 64 bits à usage généraux nommés %rax, %rbx, %rcx et %rdx, ... etc comme montré sur la figure 1.

| Nom du registre | Utilisation |
|---|---|
| rax | accumulateur, contient la valeur de retour des fonctions |
| rbx | registre général |
| rcx | compteur de boucle |
| rdx | partie haute d'une valeur 128 bits |
| rsi | adresse source pour déplacement ou comparaison |
| rdi | adresse destination pour déplacement ou comparaison |
| rsp | pointeur de pile (stack pointer) |
| rbp | pointeur de base (base pointer) |
| r8 | registre général |
| r9 | registre général |
| r10 | registre général |
| r11 | registre général |
| r12 | registre général |
| r13 | registre général |
| r14 | registre général |
| r15 | registre général |

**Figure 1**: registres de la famille Intel 64

En plus des ces registres, les processeurs Intel 64 contiennent un compteur de programme (instruction pointer) nommé «rip»

### 1.2.1. Accès à une partie d'un registre

Il est possible de n'utiliser qu'une portion de ces registres constituée de 32 bits : dans le cas du registre %rax cette portion est nommée %eax. On peut également n'utiliser que 16 bits (les processeurs intel étaient à l'origine d'une capacité de 16 bits ; le «e» de %eax signifiant extended). De ce cas la portion du registre %rax est nommée %ax.
Les 8 bits de poids faible et ceux de poids fort de %ax sont nommés respectivement %al (low) et %ah (high).
Ce principe de décomposition de registres est illustré par la Figure 2.

| Nom du registre | Taille |
|---|---|
| rax , rbx , rcx , rdx, rdi , rsi , rbp, rsp , r8 , r9 , ... ,r15 | 64 bits |
| eax, ebx, ecx, edx, edi , esi , ebp, esp, r8d, r9d, ... ,r15d | 32 bits |
| ax, bx, cx, dx, di , si , bp, sp, r8w, r9w, ... ,r15w | 16 bits (15:0) |
| ah, bh, ch, dh | 8 bits high (15:8) |
| al , bl , cl , dl , dil , sil , bpl , spl ,r8b, r9b, ... ,r15b | 8 bits low (7:0) |

**Figure 2**: Décomposition de registres

**Remarque 1.** Ces registres ne sont pas tous indépendants : si le registre `%al` est modifié, il en est de même des registre %ax et %eax alors que le registre %ah n'est pas affecté (ainsi que les autres registres).

**Remarque 2.** Des registres 128 bits sont formés par la conjonction de 2 registres 32 bits (%rdx:%rax) et utilisés lors de certaines opérations (`mul`).

### 1.2.2. Le registre RFLAGS

Le registre rflags contient des informations concernant le résultat de l'exécution d'une instruction (il est mit à jours après l'exécution de chaque instruction). Seuls certains des 32 bits de la partie eflags sont utilisés et certains bits du registre appelés drapeaux (flags) ont une signification particulière. Les noms et significations des principaux bits de eflags sont donnés dans la figure 3.

| **Flag (bit)** | **Type** | **Signification (utilisation)** |
|---|---|---|
| CF Carry Flag (bit 0) | flag d'état | c'est le flag de retenue. Parmi les flags d'état, seul **CF** peut être modifié directement par certaines instructions qui sont:<br>• CMC (pour ***Co**mplement **C**arry Flag*) : Inverse l'état du flag.<br>• CLC (pour ***Cl**ear **C**arry Flag*): mise à zéro.<br>• STC (pour ***Se**t **C**arry Flag*) : mise à un. |
| PF Parity Flag (bit 2) | flag d'état | vaut 1 si l'octet de poids faible du résultat de la dernière opération arithmétique contient un nombre pair de bits à 1 |
| AF Auxiliary Carry Flag (bit 4) | flag d'état | vaut 1 si le résultat de la dernière opération arithmétique provoque une retenue sur le troisième bit |
| ZF Zero Flag (bit 6) | flag d'état | vaut 1 lorsque le résultat est 0 |
| SF Sign Flag (bit 7) | flag d'état | bit de signe du résultat |
| OF Overflow Flag (bit 11) | flag d'état | dépassement (le résultat trop grand) |
| DF Direction Flag (bit 10) | flag de contrôle | utilisé conjointement avec les instructions opérant sur les chaînes de caractères. Lorsqu'il est à 1, les adresses des chaînes de caractères sont auto décrémentées (allant ainsi des adresses les plus hautes vers les adresses les plus basses). Lorsqu'il est à 0, les adresses des chaînes de caractères sont auto incrémentées. Aucun résultat d'opération ne permet de modifier ce flag. Seules les deux instructions<br>CLD (pour ***Cl**ear **D**irection Flag*) et STD (pour ***Se**t **D**irection Flag*) permettent de spécifier explicitement son état. |
| TF Task Flag (bit 8) | flag système | lorsqu'il est à 1, ce flag permet le débogage en mode pas à pas. |
| IF Interrupt Flag (bit 9) | flag système | il contrôle la façon dont le processeur réagit aux requêtes d'interruptions masquables (désactivables). Lorsqu'il est à 1, le processeur peut répondre à toutes les interruptions. Lorsqu'il est à 0, le processeur ne peut répondre qu'aux interruptions non masquables. |
| IOPL I/O Privilege Level (bits 12 et 13) | flag système | |
| NT Nested Task (bit 14) | flag système | |
| RF Resume Flag (bit 16) | flag système | il contrôle la réponse du processeur aux exceptions de débogage (Il assure par exemple que le débogage en pas à pas n'intervient qu'une seule fois par instruction) |
| VM Vitrual 8086 Mode (bit 17) | flag système | avec ce flag à 1, le processeur est en mode virtuel 8086. Il revient en mode protégé sinon. |
| AC Alignement Check (bit 18) | flag système | indique si une vérification d'alignement des références mémoire sera effectuée ou pas |
| VIF Virtual Interrupt Flag (bit 19) | flag système | |
| VIP Virtual Interrupt Pending (bit 20) | flag système | |
| ID Identification Flag (bit 21) | flag système | si flag peut être modifié par un programme, alors le processeur supporte l'utilisation de l'instruction CPUID |

**Figure 3**: utilisation du registre rflags

**Remarque**: **l**es bits 63 à 32, 31 à 22, 15, 5, 3, 1 sont des bits réservés par le constructeur. Leur utilisation et fonctionnement restent inconnues et impossibles aux programmeurs.

**Instructions manipulant le registre RFLAGS**

En plus des instructions permettant de modifier un ou plusieurs bits du registre RFLAGS, certaines instructions permettent de lire ou d'écrire tout ou partie du registre RFLAGS.

- LAHF : les bits 0 à 15 de RFLAGS (partie FLAGS) sont mis dans le registre AH (opération de lecture).
- SAHF : le contenu du registre AH est placé dans les bits 0 à 15 de RFLAGS (opération d'écriture).
- POPF : le mot de 16 bits actuellement sur le haut de la pile est placée dans les bits 0 à 15 de RFLAGS (opération d'écriture) - en mode 64 bits, l'instruction se comporte comme POPFQ.
- POPFD : le double mot (32 bits) actuellement en haut de la pile est placé dans RFLAGS (opération d'écriture) - en mode 64 bits, l'instruction se comporte comme POPFQ.
- POPFQ : le quadruple mot (64 bits) actuellement en haut de la pile est placé dans RFLAGS (opération d'écriture).
- PUSHF : les bits 0 à 15 de RFLAGS sont poussés sur la pile (opération de lecture) - en mode 64 bits, l'instruction se comporte comme PUSHFQ.
- PUSHFD : le contenu de RFLAGS (32 bits) est poussé sur la pile (opération de lecture) - en mode 64 bits, l'instruction se comporte comme PUSHFQ.
- PUSHFQ : le contenu de RFLAGS (64 bits) est poussé sur la pile (opération de lecture).