32 ou 64 bits ?

Depuis la généralisation des processeurs Intel de la gamme Core 2 et AMD 64, les utilisateurs ont le choix entre des systèmes d'exploitation fonctionnant en mode 32 bits ou en mode 64 bits. Concrètement, sur quoi portent les différences ?

Lorsque le fondeur Intel lance, en 1986, son nouveau processeur 80386 (ou i386), il le dote, pour la première fois, d'un bus d'adresses sur 32 bits, c'est-à-dire de la possibilité de référencer jusqu'à quatre milliards d'octets différents, une quantité qui, à l'époque, paraît fabuleuse : rares sont alors les micro-ordinateurs qui dépassent les quelques mégaoctets de mémoire.

Vingt ans après, la situation s'est totalement inversée : les serveurs, stations de travail et maintenant les ordinateurs personnels approchent, atteignent ou dépassent la limite fatidique des 4 Gio *1*. Heureusement, entre temps, les fondeurs, à commencer par AMD, se sont aperçus que l'adressage 32 bits n'allait plus suffire. Ils ont donc à nouveau doublé la capacité d'adressage pour la passer à 64 bits *2*. Toutefois, cette transition s'est faite silencieusement, puisque l'activation du mode 64 bits se fait logiciellement et n'est pas activée par défaut à l'initialisation du processeur, qui continue donc, par défaut, à fonctionner comme un « bon vieux » 32 bits.

De fait, à part les administrateurs de serveurs importants, qui utilisent systématiquement le mode 64 bits, la plupart des utilisateurs, même professionnels, ignorent cette possibilité, aidés en cela par la diffusion quasi-exclusive de version 32 bits des systèmes d'exploitation Microsoft *3*. Là dessus s'ajoute un ensemble de « rumeurs » selon lesquels les versions 64 bits n'apportent rien ou pis, dégradent les performances, que la plupart des pilotes ne fonctionnent pas en 64 bits, etc. ; bref, qu'il vaut mieux les éviter, tant que l'on excède pas 4 Gio de mémoire. Or, il est impossible aux versions 32 bits de Windows® Vista™ ou XP™ de gérer l'intégralité des 4 Gio de RAM avec lesquels certaines machines sont livrées.

L'illusion des 4 Gio

La première raison de cette limitation n'est pas liée au système d'exploitation. Elle résulte tout simplement de l'existence de périphériques, dont les adresses doivent elles aussi prendre place en-deça de 4 Gio. Les contrôleurs USB, Ethernet, Firewire, Wi-Fi, Bluetooth, le chipset lui-même, contiennent des registres qu'il faut pouvoir lire ou écrire ; ils consomment donc des ressources mémoire ; en général, pour simplifier les allocations, toute une partie de l'espace mémoire est consacrée au dialogue avec les périphériques (souvent situés sur des bus PCI). Par conséquent, la zone de RAM se trouvant à la même adresse disparaît (effet de masque). Le BIOS, cette partie du système écrite en mémoire morte (Flash) et responsable du démarrage de la machine, s'y trouve également logé.

Il ne s'agit là que de quelques kilo-octets sans importance. Cependant, cet effet de masque prend de l'ampleur avec les cartes vidéo modernes : il n'est pas rare de rencontrer des cartes équipées de 256, 512 voire 1 024 Mio de mémoire. Dans tous les cas, cet espace mémoire est accessible par le processeur via un bus de type PCI Express ; dès lors, il masque né-

1. La nouvelle norme ISO impose d'utiliser comme abbréviations pour les préfixes multiplicatifs des unités informatiques les formes standard postfixées par « i » : kiO (kilo-octet), MiO (méga-octet), GiO, etc. afin de distinguer la valeur standard (multiple de 1000) de la valeur « informatique » (multiple de 210 = 1024).
2. Cette transition s'est faite directement chez AMD. En revanche, Intel a tout d'abord crée une architecture particulière IA64 (processeur Itanium®) avant de se raviser et de fusionner ses deux lignes de produit 32 et 64 bits en un silicium unique.
3. Les systèmes d'exploitation libres, comme Linux ou les différents BSD offrent tous le choix entre des versions 32 ou 64. Mac OS X (Leopard), dérivé de FreeBSD et MachOS, tourne en mode 64 bits.

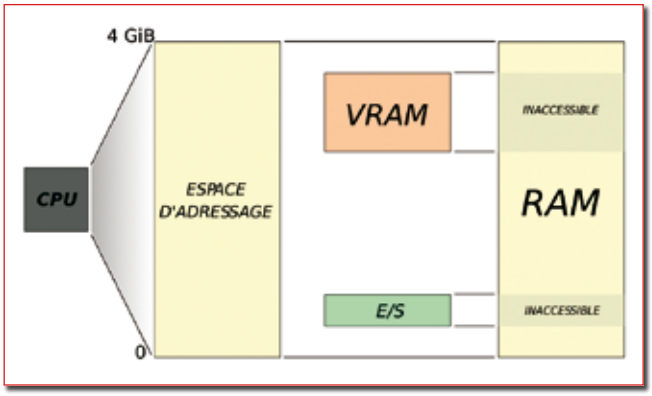


Figure 1 : L'effet de masque. Sur un espace d'adressage de 4 GiO, la place mémoire occupée par les périphériques et par la RAM vidéo masquent la RAM principale située à la même adresse, qui devient inaccessible : le processeur ne peut donc utiliser l'intégralité des 4 GiO de RAM principale.

cessairement la même quantité de mémoire vive principale. Ainsi, équipé d'une carte graphique de 1 GiO, un ordinateur muni d'un système d'exploitation 32 bits ne peut utiliser plus de 3 GiO de RAM : « voir ou produire, il faut choisir ».

Mais ce n'est pas tout. À cette limite, que l'on pourrait qualifier de matérielle, vient se greffer une limite logicielle, liée au système d'exploitation. Pour bien comprendre de quoi il s'agit, il convient de s'attarder quelque peu sur le fonctionnement intime du microprocesseur, particulièrement sur la façon dont il gère la mémoire.

MMU et la mémoire perdue

À chaque instant, sur un système d'exploitation multitâche préemptif, comme le sont tous les systèmes modernes, tournent plusieurs programmes : les applications utilisateurs et le noyau, c'est-à-dire le système d'exploitation lui-même. Pour assurer un minimum de protection entre ces tâches (éviter que l'on vienne, par erreur, écrire dans les données du voisin), le système utilise un mécanisme spécial de gestion de la mémoire, appelé mémoire virtuelle.

Essentiellement, chaque programme évolue dans un espace mémoire autonome, appelé espace logique ou virtuel. À chaque accès mémoire, une structure spéciale du microprocesseur, la MMU (Memory Management Unit), ajoute, de manière transparente au logiciel, un décalage à l'adresse logique de façon à former la véritable adresse physique. Ce décalage peut ou non être variable : avec la MMU, des adresses logiques apparemment contiguës peuvent en fait se trouver très loin l'une de l'autre en mémoire physique (voir figure 2). La MMU permet également, par une méthode complexe, d'utiliser de l'espace disque pour augmenter la mémoire disponible : ainsi, même avec 2 GiO de RAM, les applications peuvent utiliser 3 GiO ou plus, le supplément de mémoire étant placé dans un fichier particulier appelé fichier d'échange ou swap [4].

Pour effectuer ses translations, la MMU obtient la valeur du décalage à appliquer à chaque accès en consultant une structure arborescente résidant elle-même dans une zone de la mémoire. Pour des questions d'efficacité, les valeurs les plus fréquentes sont répliquées dans un cache interne, le TLB (Translation Lookaside Buffer) : si la MMU devait effectuer plusieurs accès mémoire (nécessaires au parcours de la structure arborescente) pour récupérer le décalage correct à chaque instruction du CPU, la vitesse de ce dernier serait drastiquement réduite.

Ce cache doit être systématiquement vidé à chaque changement de tâche, puisque chacune possède ses propres espaces logiques et physiques, donc ses propres valeurs de translation. Cela se produit généralement tous les centièmes de seconde, n'occasionnant qu'une légère perte de performances [5], acceptable et inévitable.

Mais c'est également le cas lorsqu'un programme fait appel à un service du noyau : l'exécution se trouve transférée de la tâche utilisateur vers le système d'exploitation, qui dispose également de son espace logique d'adressage. Or, ces appels systèmes sont nombreux et fréquents, puisqu'ils concernent tous les accès disques et périphériques en général, les primitives graphiques, les demandes de blocs de mémoire, le lancement et la terminaison des programmes utilisateurs, etc. S'il fallait vider le cache à l'occasion de chaque appel système (et de chaque retour d'un tel appel), là aussi le système serait ralenti.

La solution, bien sûr, est de ne pas effectuer ce vidage. Pour cela, il faut que l'espace logique de toutes les tâches, quelles qu'elles soient, inclue l'espace logique du noyau. Autrement dit, si le système d'exploitation occupe une place de 1 GiO en mémoire physique, 1 GiO de mémoire logique, par exemple la plage allant de 3 à 4 GiO, devra être systématiquement réservée au noyau. De sorte que, lorsqu'un appel système se produira, les accès mémoire effectués par le système d'exploitation pourront se faire dans l'espace logique de la tâche. En contrepartie, 1 GiO disparaît de l'espace mémoire utilisable

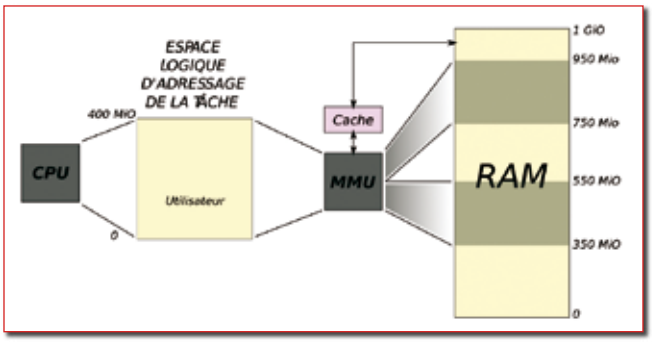


Figure 2 : Un programme occupe 400 MiO de mémoire. Dans l'espace logique, cette quantité est vue comme un seul bloc situé entre les adresses 0 et 419 430 400. Mais dans l'espace physique « réel », cette mémoire est allouée comme deux blocs de 200 MiO séparés par un « trou ». La MMU prend en charge la transformation des accès logiques vers les accès physiques correspondants, en consultant un arbre de translation stocké en mémoire, dont une partie est répliquée dans un cache pour améliorer la performance.

4. En toute rigueur, il est possible d'utiliser ce système de swap pour compenser la perte de mémoire due à l'effet de masque. Cependant, les accès disques sont en moyenne 100 000 à 1 000 000 de fois plus lents que les accès mémoire (10 ms contre 10 ns) !
5. Plus le système commute souvent et plus cette perte est importante. Il faut choisir entre l'« illusion du multitâche » et l'efficacité.

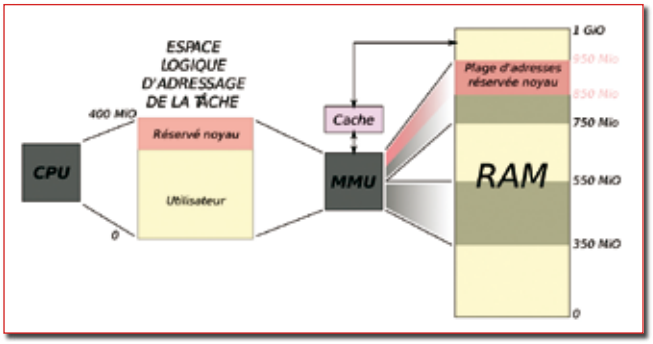


Figure 3 : Sur l'espace mémoire logique de 400 MiO, 100 MiO doivent être réservés au noyau pour éviter le changement de contexte mémoire lors d'un appel système. Ces 100 MiO occupent les mêmes adresses logiques et physiques, quelle que soit la tâche en cours d'exécution.

par les applications utilisateurs, qui ne peuvent plus utiliser que 3 GiO de RAM *6* *7*.

Windows™ n'est pas le seul système d'exploitation à troquer mémoire contre efficacité. C'est aussi le cas d'au moins un système libre, NetBSD en version 32 bits. En revanche, Linux 32 ou Mac OS X ne le font pas, préférant conserver la possibilité d'allouer l'intégralité des 4 GiO à chaque tâche, en échange du ralentissement dû au vidage du TLB à chaque appel système.

Les avantages du 64 bits...

Avec un système d'exploitation 64 bits *8*, les deux problèmes évoqués jusqu'ici disparaissent. Pourquoi ? Premièrement, parce que la plage consacrée aux entrées-sorties et à la RAM vidéo peut être déportée largement au-delà de la fenêtre des 4 GiO de RAM principale. Il n'y a plus de conflit d'adresses, donc plus de RAM masquée *9*.

De même, l'espace mémoire fixe bloqué par le noyau n'a plus à se trouver dans l'espace logique dédié à la RAM principale. Les 4 GiO sont donc de nouveau logiquement adressables *10*.

Outre mettre à disposition plus d'espace mémoire, le modèle 64 bits offre des performances accrues, aussi bien pour des applications « standard » que des applications gourmandes en temps de calcul.

En effet, en mode 64 bits, les processeurs non seulement voient la largeur de leurs registres *11* internes doubler (de 32 à 64 bits), mais également leur nombre. Les compilateurs disposent de seize registres entiers *12* de 64 bits, et seize registres flottants de 128 bits au lieu de huit registres entiers de 32 bits et huit registres flottants : ils peuvent alors optimiser le code machine en réduisant les appels à la mémoire.

Passer d'un code 32 bits à un code 64 bits devrait en principe ne requérir qu'une simple recompilation de la part de l'éditeur, ou de l'utilisateur dans le cas du logiciel libre. Ce n'est toutefois qu'un vœux pieux : de mauvaises pratiques de programmation (particulièrement la conversion entre pointeurs et entiers) entâchent le code 32 bits et le rendent impossible à porter tel quel en 64 bits. Ce genre de « bricolage » logiciel est fréquent dans les pilotes de périphériques, ce qui explique qu'à l'heure actuelle, rares sont les pilotes qui fonctionnent correctement sous Windows Vista® ou XP® 64 bits.

Conscients de la difficulté que la migration poserait, les fondeurs ont pourvu leurs processeurs d'un mode de compatibilité. En l'utilisant, il est possible de faire tourner des applications 32 bits sur son système d'exploitation 64 bits. Des tests, effectués par AMD, prouvent que les applications 32 bits exécutées en mode « 64 bits compatible » n'enregistrent qu'une perte de performance négligeable par rapport à leur exécution en mode 32 bits « réel ». Ce mode « compatible » ne s'applique cependant pas aux pilotes, qui sont des parties intégrantes du noyau. En revanche, les applications mettant en jeu de nombreuses opérations mathématiques, comme de l'analyse EF, tournent plus vite en mode 64 bits qu'en mode 32 bits.

... et ses inconvénients !

La situation n'est là encore pas aussi manichéenne. Si le 64 bits apporte un gain de

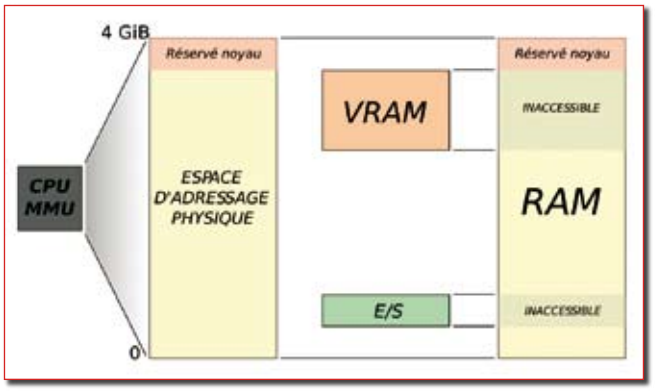


Figure 4 : Situation globale. Sur les 4 GiO de RAM, seule la partie jaune est effectivement disponible pour les applications.

6. Il est impossible de compenser cette perte de mémoire virtuelle par une allocation d'espace sur le disque dur. Le swap permet de masquer un manque de mémoire physique, mais pas de mémoire logique.
7. Sous Windows™, l'allocation par défaut est de 2 GiO au système et 2 GiO à l'utilisateur. Ce dernier peut cependant repousser la limite à 1 GiO / 3 GiO.
8. 64 bits est un terme abusif, qui ne se réfère qu'à la taille des registres internes du processeur (voir plus loin). En réalité, la MMU ne travaille que sur des adresses de 48 bits, soit 256 TiO. En outre, ces 48 bits ne sont pas tous présents à l'extérieur du boîtier. Les processeurs Core 2 de chez Intel, par exemple, ne disposent que de 36 bits d'adresses, soit 64 GiO adressables.
9. Ceci n'est rigoureusement vrai qu'à la condition que le chipset associé au processeur supporte un adressage plus large que 32 bits, ce qui, chez Intel, ne date que de la version Santa Rosa du chipset.
10. En réalité, le noyau a toujours besoin de mémoire. La quantité de mémoire physique maximale disponible pour l'utilisateur n'a donc pas augmenté lors de la transition 32 vers 64. Simplement, on dispose en mode 64 bits de la possibilité d'allouer du swap, alors que cette option n'était pas disponible en mode 32 bits, en raison du chevauchement des espaces logiques.
11. Les registres sont des unités de mémoire interne sur lesquels les instructions élémentaires du CPU travaillent. Leur temps d'accès est nul. Ils peuvent contenir des opérandes, des résultats ou bien encore des pointeurs mémoire. Plus ils sont nombreux, et moins le processeur a besoin d'échanger avec la mémoire pour stocker des résultats intermédiaires, ce qui accroît la vitesse d'exécution.
12. Un registre entier représente un nombre entier ou un pointeur. Un registre flottant stocke la valeur d'un nombre réel, ici en double précision.

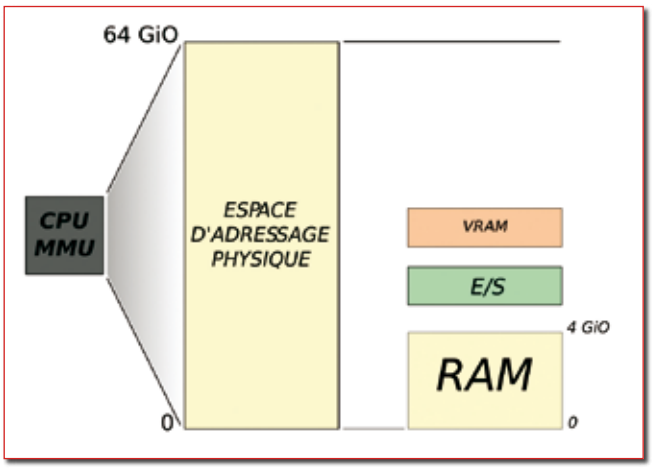
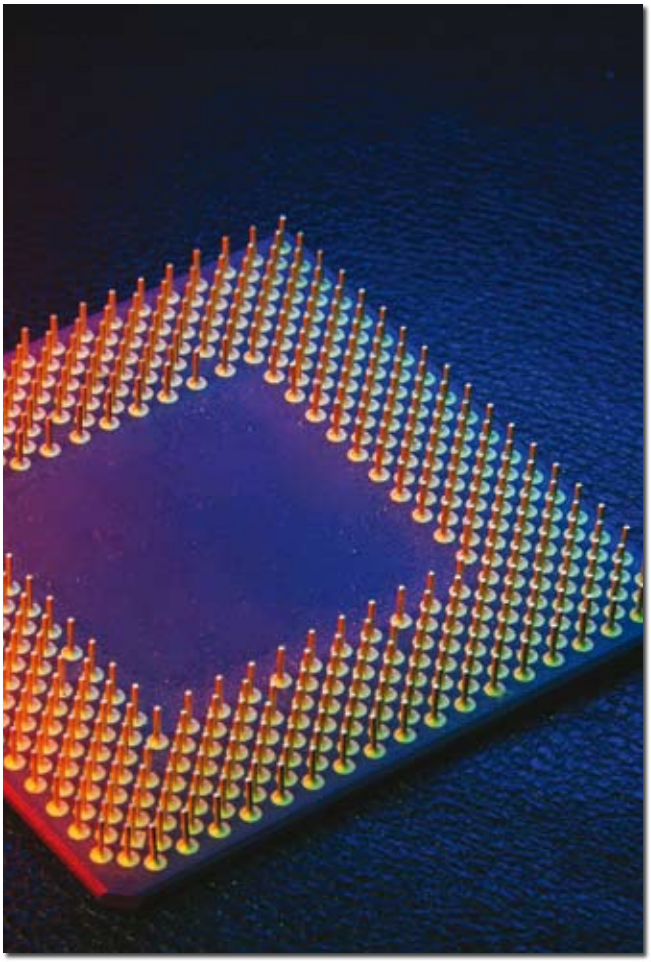


Figure 5 : Avec un adressage sur « 64 bits » (en réalité 48 au mieux), les zones réservées aux entrées-sorties et à la RAM ne se chevauchent plus.

performance, il entraîne aussi un certain nombre d'inconvénients. En premier lieu, la taille des programmes augmente. En effet, chaque adresse stockée dans le code prend désormais huit octets au lieu de quatre. En outre, les registres 64 bits, lorsqu'il faut les sauver en mémoire, prennent deux fois plus de place que leurs contreparties 32 bits.

La capacité « réelle » du cache de données est ainsi pratiquement divisée par deux, et, dans certain cas, la performance s'en ressent (des études montrent que l'efficacité du cache diminue de 5 %) : certains programmes tournent moins vite quand ils sont compilés en mode 64 bits qu'en mode 32 bits. Toutefois, ces contre-performances devraient progressivement disparaître, les nouvelles versions des compilateurs optimisant davantage le code pour les plates-formes 64 bits que leurs prédécesseurs.



Conclusion

Comme on le voit, il y a du pour et du contre à passer en 64 bits. À l'heure actuelle, la plupart des utilisateurs de Windows™ restent en mode 32 bits, puisqu'il s'agit de la version livrée en standard, que ce soit Vista® ou XP®. C'est d'ailleurs la seule qui soit compatible avec certains utilitaires comme Flash™, par exemple. Il n'y a d'ailleurs aucune véritable raison de passer en 64 bits tant que l'on se contente d'utiliser de modestes quantités de mémoire, et de faibles ressources mathématiques. À l'inverse, dès qu'il s'agit de manipuler de grandes quantités de données, comme dans le domaine de la CAO, du calcul éléments finis (là où précisément les cartes graphiques disposent de quantité de RAM importantes qui majorent l'effet de masque) ou du calcul mathématique en général, le passage en 64 bits devrait apporter de nettes améliorations, du fait non seulement de la possibilité de disposer de davantage de mémoire, mais également des registres supplémentaires.

Il n'en reste pas moins que – pour ceux qui n'utilisent ni système d'exploitation ni logiciels libres ***13*** – le passage en 64 bits suppose l'acquisition de la version 64 bit du système et de l'application. Il faut donc que cette dernière existe, ce qui est loin d'être le cas chez l'ensemble des éditeurs. Migrer sur une plate-forme 64 bits et exécuter les applications en mode compatible constitue la meilleure solution en attendant une généralisation (à moyen terme) de la technologie. Éternel problème de la poule et de l'œuf : tant qu'il y aura peu de demande en 64 bits, les éditeurs ne proposeront pas de versions de leurs logiciels adaptés ; inversement, tant que celles-ci n'existeront pas, pourquoi les utilisateurs dépenseraient-ils leur argent pour rien ?

Vincent Habchi

Brève Bibliographie virtuelle

http://chemnitzer.linux-tage.de/2008/vortraege/shortpaper/64bitLinux_CLT08.pdf

http://www.x86-secret.com/articles/cpu/32vs64/32vs64.htm

http://www.appleinsider.com/articles/08/09/03/road_to_mac_os_x_snow_leopard_64_bits_santa_rosa_and_the_great_pc_swindle.html (série de quatre articles principalement axés sur le prochain système d'exploitation d'Apple™).

13. Ni MacOS X Leopard…