Virtualisation

1. Préambule

L'objectif de ce TP est de découvrir des mécanismes bas-niveau de la virtualisation, assistée par des instructions spécifiques, au travers de la documentation constructeur et d'un hyperviseur minimal.

2. Prérequis

A minima, un navigateur avec un accès Internet (avec la fonction de lecture PDF). Un éditeur de texte orienté développement ou un environnement de développement intégré peut être un plus pour ceux qui sont à l'aise avec ce type d'outil.

3. Instructions matérielles pour la virtualisation

Exercice 3-1

Objectif : appréhender les instructions AMD dédiée à la virtualisation

Récupérer les manuels AMD suivant sur la page dédiée aux développeurs :

- AMD64 Architecture Programmer's Manual Volume 2: System Programming
- AMD64 Architecture Programmer's Manual Volume 3: General Purpose and System Instructions

Dans le manuel AMD volume 3, identifier dans le tableau D-2 (annexe D.3, à partir de la page 573/682) les instructions de virtualisation suivantes et noter le niveau de privilège nécessaire (CPL) :

Mnemonic	Description	CPL
CLGI	Clear Global Interrupt Flag	
INVLPGA	Invalidate TLB Entry in a specified ASID (Address Space IDdentifier)	
VMLOAD	Load State from VMCB	
VMMCALL	Call VMM	
VMRUN	Run Virtual Machine	
VMSAVE	Save State to VMCB	
STGI	Set Global Interrupt Flag	

Compte-tenu du niveau CPL de ces instructions, sont-elles prévues pour être utilisée dans du code noyau ou du code utilisateur ?

Le VMCB (Virtual Machine Control Block) peut être vu comme le pendant du VMCS que l'on retrouve dans la documentation Intel. La structure d'un VMCB est décrite dans le manuel AMD volume 2 dans l'annexe B (p647/688). Elle permet notamment de définir la liste des opérations que l'hyperviseur doit intercepter (tableau B-1) et de sauver l'état des informations sur la VM (tableau B-2).

Rappel : l'interception par l'hyperviseur interrompt l'exécution de la VM (à l'image d'un processus qui est interrompu par l'ordonnanceur de processus). Cet événement est appelé #VMEXIT (décrit dans le chapitre 15.6 du manuel AMD volume 2).

La liste des codes #VMEXIT est donnée dans l'annexe C du manuel AMD volume 2.

Quel peut-être l'intérêt d'intercepter l'écriture du registre CR3 effectuée au sein d'une VM (cf. chapitre 5 du manuel AMD volume 2 sur la pagination) ?

La pagination imbriquée (cf. chapitre 15.25.1 du manuel AMD volume 2) introduit une seconde structure de tables de page, appelée NPT (conceptuellement similaire au mécanisme EPT des processeurs Intel). Est-ce que, avec l'activation de ce mécanisme, l'interception de l'écriture du registre CR3 conserve un intérêt ?

Quel peut-être l'intérêt d'intercepter l'exécution de l'instruction CPUID (cf. annexes E et D du manuel AMD volume 3 sur le rôle de CPUID) au sein d'une VM ?

Quel peut-être l'intérêt d'intercepter l'écriture du registre IDTR effectuée au sein d'une VM (cf. chapitre 8 du manuel AMD volume 2 sur les exceptions et les interruptions) ?

Exercice 3-2

Objectif : étudier le code source d'un hyperviseur minimale utilisant les instructions AMD SVM dédiée à la virtualisation

Le code source du projet SimpleSvm cible un système Windows et est implémenté sous forme d'un pilote noyau à compiler avec Visual Studio. Afin de réduire la quantité de code assembleur, il fait appel à des fonctions C, nommée fonctions intrinsèques, qui correspondent à des instructions assembleur :

- Intrinsics Available on All Architectures
- x64 (amd64) Intrinsics List

Les instructions de virtualisations AMD commencent par __svm, (à l'exception de vmmcall qui n'est pas disponible de cette manière).

Y a-t-il un intérêt à implémenter cet hyperviseur minimaliste sous forme de pilote ?

Le projet est constitué des fichiers suivants :

- SimpleVM.cpp contient le code du pilote, notamment du code lié à la structure d'un pilote Windows :
  - la fonction DriverEntry() est le point d'entrée du pilote (à l'image d'une fonction main() dans un programme C classique), elle réalise certaines opérations itinérantes à un pilote Windows avant d'appeler la fonction SvVirtualizeAllProcessors() ;
  - la fonction SvDriverUnload() est appelée lors de l'arrêt du pilote (déclenché par l'administrateur ou lors de l'arrêt du système) ;
  - la fonction SvPowerCallbackRoutine() est liée à la mise en veille et présente peu d'intérêt pour l'exercice;
- SimpleSvm.hpp contient notamment la déclaration de la structure VMCB ;
- x64.asm contient la procédure SvLaunchVm qui met en place une boucle qui démarre la VM (vmload & vmrun), traite les #VMEXIT (vmsave et appel à SvHandleVmExit()) et reprend l'exécution de la VM (retour au début de la boucle) ou prépare l'arrêt de l'hyperviseur (sortie de la boucle).

SimpleSvm est-il un hyperviseur de type 1 ou de type 2 ?

Est-ce que SimpleSvm intercepte les tentatives d'écriture du registre CR3 ? Pourquoi ?

Quels sont les motifs de #VMEXIT traités par SimpleSvm au sein de la fonction SvHandleVmExit() ?

Que se passe-t-il en cas d'exécution de l'instruction vmrun par une VM ?

Comment se comporte SimpleSvm pour les autres motifs ?

Quel est le nom du constructeur du processeur renvoyé par l'instruction CPUID quand SimpleSvm est démarré ?

Quelle instruction peut être utilisée par une VM pour désactiver SimpleSvm ?