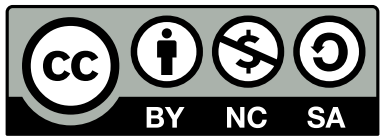
1011 - Entrées-sorties

INF2171
Organisation des ordinateurs et assembleur

Jean Privat

Université du Québec à Montréal

INF2171 - Organisation des ordinateurs et assembleur
v233



Rappel

Architecture

- UCT, mémoire, périphériques (entrées-sorties), bus

Boucle d'exécution de l'UCT

- Lire l'instruction en mémoire (*fetch*)
- Décoder l'instruction
- Exécuter l'instruction
- Incrémenter le compteur ordinal (`pc`)

Plan

Vocabulaire des périphériques

Périphérique

- Dispositif physique pour faire des entrées sorties
- Moniteur, clavier, souris, caméra, haut-parleur, microphone, etc.

Contrôleur

- Dispositif électronique dans l'ordinateur
- Connecté au bus d'un coté et au périphérique de l'autre
- Carte d'extension, puces sur la carte mère, ou intégré dans le processeur

Vocabulaire des périphériques (suite)

Microcontrôleur

- Dispositif électronique dans le périphérique
- Le périphérique et quasiment un mini-ordinateur autonome
- Déporte de la complexité dans des circuits du périphérique

Pilote

- Programme gérant le périphérique
- Écrit en langage de bas niveau (C, assembleur...)
- Exécuté par l'UCT (processeur principal)
- Habituellement un morceau du système d'exploitation (cf. INF3173)

Entrées-sorties programmés

Périphériques en mémoire

- *Memory-mapped I/O*, MMIO
- Les périphériques sont sur le bus
- Ils peuvent répondre aux demandes de lectures et écritures « mémoire » du processeur

Adresses des périphériques

- Décidés par le fabriquant
  - Exemple: StarFive JH-7110 Technical Reference Manual
- Négociés par le bus et le périphérique (auto-configuration)
  - PCI (*Peripheral Component Interconnect*)
- Linux:
  - `sudo cat /proc/iomem`
  - `sudo lspci -v` (champs `Memory at`)

Architectures alternatives

Port-mapped I/O (PMIO)

- Les périphériques et la mémoire sont possiblement disjoints
- Instructions spéciales de l'UTC pour l'accès aux périphériques
- Plus rare / ancien
  - RISC-V ne le propose pas
  - x86: permet les deux (instructions dédiées `in` et `out`)
  - Mais légataire en x86_64
- Linux:
  - `sudo cat /proc/ioports`

Registre de périphériques

- Un périphérique émet de l'information / reçoit de l'information
- « **Registre de données** » (*data register*)
  - Morceau d'information qui peut être émise et/ou reçue
  - Ne pas confondre avec les registres du processeur
- Chaque registre de donnée a
  - Un rôle spécifique (dépendant du matériel en question)
  - Une taille (en octets)
  - Un mode d'accès (lecture/écriture/les deux)
  - Mais les règles peuvent changer dynamiquement
- Important: ne fonctionne pas comme la mémoire
  - Écrire $\rightarrow$ pour envoyer de l'information
  - Lire $\rightarrow$ pour recevoir de l'information
    Lire peut avoir un effet (consommer l'information)

Asynchronisme

- Un périphérique est (souvent) plus lent que l'UTC
- L'UCT doit **attendre** que le périphérique soit prêt
  - à recevoir des données
  - à émettre des données
- Solution: registre de contrôle (ou status)
  - Permet de connaitre l'état du périphérique
    Est-il prêt à émettre ou à recevoir ?
  - Voire permet de configurer le périphérique

Exemple RARS: Keyboard and Display MMIO

Périphérique clavier et écran simulé

Clavier

- 0xFFFF0000 (mot): registre de contrôle (lecture seule)
  - dernier bit $= 1 \rightarrow$ il y a un caractère à lire
  - dernier bit $= 0 \rightarrow$ il n'y a rien à lire
- 0xFFFF0004 (mot): registre de donnée (lecture seule)
  - dernier octet $\rightarrow$ code ASCII du caractère
  - Ne lire que si le contrôle dit `1`
  - Lire consomme le caractère (et passe le contrôle à 0)

Exercice `kbd.s`

- Lire des caractères du clavier simulé
- Et les afficher dans le terminal (`PrintChar`)
- Note: l'écriture sur le terminal simulé semble ne pas fonctionner (bug)

Affichage graphique

Écran d'ordinateur (moniteur)

Le périphérique d'affichage

Résolution

- Précision de l'affichage
- Nombre de **pixels** (*picture element*)
- VGA (1987) 640x480 (307 kpixels)
- Full HD: 1920×1080 (2.07 Mpixels)
- UHD 4K: 3840×2160 (8.29 Mpixels)

Taux de rafraichissement

- Donne l'illusion du mouvement
- $\approx$ 60Hz classiquement de nos jours (ou plus)

Carte graphique (carte vidéo)

Le contrôleur de l'écran

Coprocesseur graphique

- *Graphical Processing Unit* (GPU)
- Prend en charge des calculs spécifiques
- Très sophistiqués de nos jours
- Cours INF5071 - Infographie

Mémoire vidéo

- Dédiée: mémoire spécifique sur la carte graphique
  Partagée: une partie de la RAM classique
- Stocke les données à traiter ou convertir
- Stocke les images traités à afficher
- Plusieurs Go de nos jours

Framebuffer

- Tableau à deux dimensions en mémoire des pixels à afficher
- Nombreuses variations du codage des pixel
- Classiquement de gauche à droite et de haut en bas
  - (0,0) est en haut à gauche

RGB 24 bits (3 octets)

- 8 bits par composante rouge, vert et bleu
  $\rightarrow$ 16 millions de couleurs
- Codage répandu des couleurs (CSS du web par exemple)
- Classiquement représenté en hexadécimal
  - `0xFF0000` $\rightarrow$ rouge, `0x00FFFF` $\rightarrow$ cyan, `0xFFFFFF` $\rightarrow$ blanc, `0x000000` $\rightarrow$ noir, `0x808080` $\rightarrow$ gris, `0xF0E68C` $\rightarrow$ kaki

Exercice

- Quelle est la taille minimale d'un framebuffer à 16 millions de couleurs 3840$\times$2160 (UHD 4K)

Framebuffer

- Tableau à deux dimensions en mémoire des pixels à afficher
- Nombreuses variations du codage des pixel
- Classiquement de gauche à droite et de haut en bas
  - (0,0) est en haut à gauche

RGB 24 bits (3 octets)

- 8 bits par composante rouge, vert et bleu
  $\rightarrow$ 16 millions de couleurs
- Codage répandu des couleurs (CSS du web par exemple)
- Classiquement représenté en hexadécimal
  - `0xFF0000` $\rightarrow$ rouge, `0x00FFFF` $\rightarrow$ cyan, `0xFFFFFF` $\rightarrow$ blanc, `0x000000` $\rightarrow$ noir, `0x808080` $\rightarrow$ gris, `0xF0E68C` $\rightarrow$ kaki

Exercice

- Quelle est la taille minimale d'un framebuffer à 16 millions de couleurs 3840×2160 (UHD 4K)
  - $3840 \times 2160 \times 3 \approx 24\text{Mio}$

Exemple RARS: Bitmap Display

Périphérique graphique simulé

- Un mot (32 bits) par pixel (on gaspille un octet)
- Adresse de base et dimensions configurables

Exercice `display.s`

- Mettre un pixel blanc aux coordonnées (6,5)
- Tracer une ligne bleue de (10,10) à (21,10)
  (faire une boucle)
- Dessiner une maison rouge avec un toit vert (en lab)

Interruptions

Problème des entrée-sorties programmées

Attente active

- ou scrutation ou *polling*
- L'UCT doit interroger régulièrement les périphériques s'ils ont quelque chose à émettre
- L'UCT doit attendre que les périphériques soient prêts à recevoir
- $\rightarrow$ Temps UCT perdu à rien faire

Interruptions

Idée: permettre la suspension des programmes

- Agir quand un évènement intervient / une situation survient

Interruption matérielle

- Évènement causé par un périphérique
- Asynchrone: survient n'importe quand
- $\rightarrow$ Un paquet réseau arrive, la souris a bougé, etc.

Interruption logicielle

- Situation causée par une instruction du programme machine
- Synchrone: survient à l'exécution de l'instruction
- On parle aussi d'**exception** ou de **faute**
- $\rightarrow$ Instruction invalide, erreur d'accès mémoire, etc.
- Inclue généralement les appels systèmes (`ecall`, `ebreak`, etc.)

Interrompre le programme

Mettre en pause l'exécution du programme

- Exécuter une **routine de gestion d'interruption**
- *Interrupt handler*, *interrupt service routine* (ISR)
- Automatiquement
- À n'importe quel moment (dans le cas asynchrone)

Transparent pour le programme

- Revient à l'instruction interrompu
- Le système (registres, etc.) « semble » inchangé

Rôle prépondérant de l'unité centrale de traitement

- $\rightarrow$ Nécessite un support explicite de l'architecture

Détection d'interruption matérielle

Registre d'interruptions matérielles

- On numérote les catégories d'interruptions
  - Liste fournie par le fabricant, ou autre
- On associe un bit à chacune
  - Cela forme le registre d'interruption
- On permet à chaque périphérique de modifier son bit
  - 0 = rien à signaler / 1 = interruption demandée
  - IRQ: *Interrupt ReQuest*

Fonctionnement

- Après l'exécution d'une instruction
  l'UCT vérifie le registre d'interruption
- Si pas d'interruption, alors l'UCT charge l'instruction suivant (boucle d'exécution classique)
- Sinon l'UCT exécute la routine de gestion de l'interruption

Exemple de catégories d'interruption

IBM PC/XT (historique, 1983)

- IRQ 0 : Horloge système
- IRQ 1 : Clavier
- IRQ 2 : Second contrôleur d'interruption (PC/AT)
- IRQ 3 : Port série (COM2/COM4)
- IRQ 4 : Port série (COM1/COM3)
- IRQ 5 : Disque dur
- IRQ 6 : Lecteur de disquettes
- IRQ 7 : Port parallèle (LPT1)

Linux

- `sudo cat /proc/interrupts`
- `sudo lsirq`

Masque et choix d'interruption

Masque d'interruption

- L'UCT peut ignorer certaines interruptions
- Un registre supplémentaire précise les interruptions désactivés (masque de bits)
- Les programmes initialisent ce registre pour armer et désarmer les interruptions

Choix d'interruption

- Plusieurs interruptions peuvent être signalées en même temps
- Les interruptions sont priorisées
- Exécution des interruptions dans l'ordre de priorité

Exécution d'une interruption

Gérante unique

- Une routine en mémoire attrape et gère toutes les interruptions
- Analyse programmative des situations

Vecteur d'interruption

- Tableau en mémoire contient autant de cases que de catégories d'interruption
- On place dans les cases les adresses de routines de traitement de l'interruption

Exécution d'une interruption

- Lorsqu'une interruption est détectée, l'UCT change son compteur ordinal l'adresse de la gérante
- Vu qu'on doit revenir, l'UCT sauvegarde le compteur ordinal quelque part avant (un peu comme un `jal`)

Étapes d'exécution d'un interruption matérielle

Le travail de l'UCT

- Termine l'exécution d'une instruction
- Détermine que l'interruption n est levée
- Sauvegarde pc (et parfois d'autres registres)
- Charge la gérante (ou la case n du vecteur d'interruption) dans pc
- Exécute la routine (cycles normaux d'exécution de l'UCT)
- Termine l'exécution de la routine (instruction spéciale)
  Restaure pc (et parfois d'autres registres)
- Exécute l'instruction suivante du programme interrompu

Détails techniques

Où trouver la gérante / le vecteur d'interruption ?

- Adresse fixe en mémoire dictée par le fabriquant de l'UCT
- Un registre dédié contient l'adresse

Où sauvegarder le compteur ordinal (`pc`) ?

- Registre dédié
- Adresse fixe mémoire
- Dans la pile

Qui configure la/les gérante(s)

- Le programme d'amorçage dans la ROM, le système d' 'exploitation, etc.

Variations

Masquage automatique

- Lors du traitement d'une interruption, les interruptions moins prioritaires sont automatiquement temporairement désactivées
- Cela permet à un évènement urgent d'interrompre le traitement d'un évènement moins urgent

Contrôleur d'interruption

- Un dispositif matériel peut être dédié à la gestion des files de priorités de requêtes d'interruptions
- Le PC/AT (1984) avait deux contrôleurs en cascade (16 IRQ)
- PC modernes: APIC
  - *Advanced Programmable Interrupt Controllers*
  - 256 IRQ

Interruptions en RISC-V

Terminologie

- Interruption, exception, faute (*fault*), *trap*, *abort*, *syscall*, etc.
- Pas de consensus: variations par fabricant et par auteur

Terminologie RISC-V

- **Interruption**: cas asynchrone, interruption matérielle
- **Exception**: cas synchrone, interruption logicielle
  - Inclut aussi `ecall` et `ebreak`
  - Mais n'inclut pas les exceptions flottantes (IEEE 745)
  - Mais inclut le NaN avertisseur (*signaling NaN*)
- **Trap**: traitement de l'interruption
  - Sauver `pc`
  - Brancher sur la gérante d'interruption (*trap handler*)

Interruptions en RISC-V

Mode privilégié

- Gestion complète pour les systèmes d'exploitations, les hyperviseurs et les environnements matériels dédiés (et simulés)
- RISC-V Volume 2, Privileged Specification version 20211203
- Pas de support dans RARS

Interruption utilisateur

- Extension "N" *User-Level Interrupts*
- 8 registres de contrôle et d'état (*Control & Status Register*, CSR) pour gérer et configurer les interruptions
- Instruction `uret` pour terminer la gérante d'interruption
- Instruction `wfi` (*wait for interrupt*) pour l'UCT
- Support limité dans RARS

Registre CSR de l'extension “N”

Configuration

- `ustatus`: *user status register*
  - 2 bits utiles: `0x1` activé (ou non), `0x10` interrompu (ou non)
- `uie`: masque d'interruption (*user interrupt-enable register*)
  - un bit par type d'interruption (1=armé, 0=désarmé)
- `uip`: interruptions en attente (*user interrupt pending*)
  - un bit par type d'interruption (1=en attente, 0=absent)
- `utvec`: *user trap vector*
  - Adresse de la routine de gestion (ou tableau de routines)

Registre CSR de l'extension "N"

Gestion de l'interruption

- `uepc`: *user exception program counter*
  - Compteur ordinal d'origine (sauvegardé)
- `ucause`: type d'évènement (*user trap cause*)
  - Interruption ou exception
  - Son numéro (IRQ ou numéro d'exception)
- `utval` valeur problématique (*user trap value*)
  - Information supplémentaire pour certaines exceptions
  - Adresse ou instruction problématique
- `uscratch`: registre libre (*user scratch register*)
  - Permet de travailler proprement
  - Sans corrompre les registres généraux

Exemple RARS: Timer Tool

Horloge matérielle simulée

- Connait le temps
- Permet de programmer un minuteur

MMIO et IRQ

- `0xFFFF0018` (dword) temps actuel en millisecondes
- `0xFFFF0020` (dword) temps de réveil (lève l'IRQ 4)

Exercice `timer.s`

- Calculer, horodater et afficher les termes de la suite de Fibonacci
- Écrire "Tic." toute les secondes

Exceptions RISC-V

- Interruption logicielle / synchrone
- Gérer les problèmes et répondre aux demandes spéciales
- 14 situations définies par RISC-V
- Support limité RARS, voir `Help>RISCV>Exceptions`

Exemples (avec valeurs de `ucause`)

- 0: Adresse de `pc` mal aligné
- 1: Adresse de `pc` invalide (hors domaine)
- 2: Instruction inconnue ou illégale
- 4: Lecture mémoire mal alignée
- 5: Lecture mémoire invalide
- 6: Écriture mémoire mal alignée
- 7: Écriture mémoire invalide
- 8: Appel système utilisateur (`ecall`)

Exemple RARS: exceptions

```
lb a0, 42(zero) # Address out of range
lw a0, 1(sp) # Load address not aligned
```

Exercice `exception.s`

- Configurer une gérante d'interruption dans `utvec`
- Activer les interruption utilisateur `0x1` dans `ustatus`
- Ignorer l'instruction signalée et passer à la suivante

Mode privilégié

- **Mode privilégié**: état du processeur qui accepte des instructions supplémentaires
- Les interruptions (logicielles et matérielles) changent habituellement le mode d'exécution
- Permet une gestion avancée et sécuritaire de l'ordinateur par un système d'exploitation ou un hyperviseur
- Plus de détails en INF3173

Mode privilégié en RISC-V

- The RISC-V Instruction Set Manual - Volume II: Privileged Architecture 2021, 155 pages

Accès direct à la mémoire

Accès direct à la mémoire

- *Direct Memory Access* (DMA)
- L'UCT ne sert qu'à initier et terminer le transfert
- Le contrôleur fait directement les lectures et écritures en mémoire
  - Où passe par un contrôleur dédié

Utilisations

- Besoin de bande passante mémoire périphérique
- Disque, carte graphique, carte réseau, etc.

Conclusion

Résumé

Entrée-sorties programmés MMIO

- Périphérique d'entrée: exemple clavier
- Périphérique de sortie: exemple affichage vidéo

Interruptions

- Gestion asynchrone des périphériques
- Gestion synchrone des fautes et appels système

DMA

- Pour aller plus vite

La prochaine fois

Circuits logiques

- Additionneur
- Registre mémoire

Ouverture à INF4170 Architecture des ordinateurs