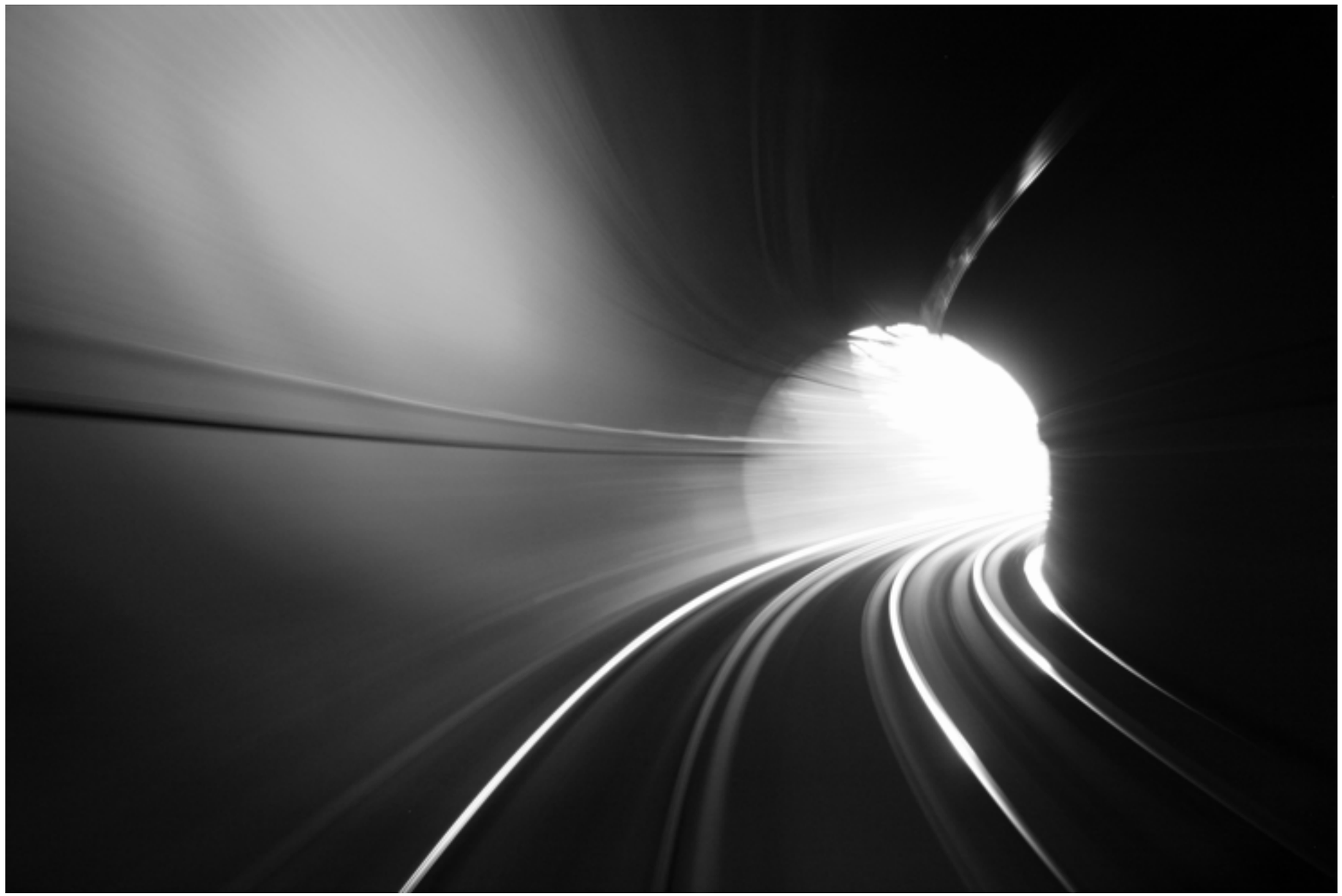
Révision mi-session



GIF-1001 Ordinateurs: Structure et Applications, Hiver 2015

Jean-François Lalonde

Examen mi-session — logistique

- Aura lieu au Vandry!
- Si votre nom de famille (alphabétiquement) est entre:
  - “Alexis” et “Lafrance”: VND-2809A
  - “Lamy” et “Yao”: VND-2811A

Examen mi-session

- Axé sur la *compréhension* et non sur le “par coeur”
- Par exemple: si on vous demande d’écrire du code en assembleur ARM, nous vous fournirons le *nom* de quelques instructions qui pourraient vous être utiles (ex: LDR, STR, MOV, etc.). À vous de savoir comment les utiliser!

Ressources

- Exercices supplémentaires et examens des années antérieures disponibles sur le site web du cours

- http://vision.gel.ulaval.ca/~jflalonde/cours/1001/h15/index.html#ressources

Format des nombres

- Entiers non-signés
- Entiers signés, notation complément-2
  - exemple: -5 en complément-2 sur 4 bits? sur 8 bits?
- IEEE754
  - pas besoin de connaître le format par coeur, mais il faut savoir l'interpréter.
- ASCII
  - comprendre comment ça fonctionne

Structure interne

- Composantes principales: ALU, CCU, mémoires
- Cycle d'instructions?
  - fetch, decode, execute!
- Comment décrit-on une mémoire?
  - adresses & taille des mots
- Bus
  - Quels sont les 3 différents types?
  - À quoi sert un décodeur d'adresses?

Instructions & micro-instructions

- De quoi est composée une instruction?
  - opcode + paramètres
- Une instruction = séquence de micro-instructions
- Utilisez le simulateur de micro-instructions!

ARM

- Connaître les éléments de base de l’architecture ARM
  - Taille des instructions?
  - Nombre de registres?
  - RISC ou CISC?
- Ok, ça c’est un peu du “par coeur”…

Assembleur ARM

- Programmes simples
- Nous vous donnerons les instructions importantes
  - Adressage, accès mémoire
  - Arithmétique
  - Conditions
  - Branchements
  - Appel de fonction
  - Pile

Assembleur ARM

- Lien entre fichier texte et contenu mémoire
  - Fichier texte: code *en texte*, écrit par nous, compréhensible pour un humain
  - Contenu mémoire: code *en binaire*, “traduit” par l’assembleur, compréhensible pour un ordinateur

Ficher texte

```
main
   MOV R1, #2
   MOV R2, #1
   ADD R0, R1, R2
   B main
```

Contenu mémoire

```
e3a01002
e3a02001
e0810002
eafffffb
```

Assembleur ARM

- Comment sont représentées les instructions suivantes?
  - `LDR R0, =maVar`
  - `B main`

Code en format texte

```
   SECTION .text : CODE (2)
   CODE32

main
   LDR R0, =maVar ; R0 = adresse de maVar
   MOV R1, #1
   STR R1, [R0]
   B main

   ; la RAM commence ici
   SECTION  `.noinit`:DATA(2)

maVar DS32 1 ; une variable de 32 bits
```

Même code en format binaire

```
00000080  e59f0008 e3a01001 e5801000 eafffffb
00000090  00100000 -------- -------- --------
000000a0  -------- -------- -------- --------
000000b0  -------- -------- -------- --------
```

Code déconstruit (disassembly)

```
   LDR R0, =maVar ; R0 = adresse de maVar
main:
         0x80: 0xe59f0008     LDR        R0, [PC, #0x8]
   MOV R1, #1
         0x84: 0xe3a01001     MOV        R1, #1
   STR R1, [R0]
         0x88: 0xe5801000     STR        R1, [R0]
   B main
         0x8c: 0xeafffffb     B          main
```

équivalent à: `B #-20`

Interruptions

- Fonctionnement, utilité
- Table des vecteurs d’interruption
- Différentes priorités