Ports et mode texte VGA

2017 – 2018

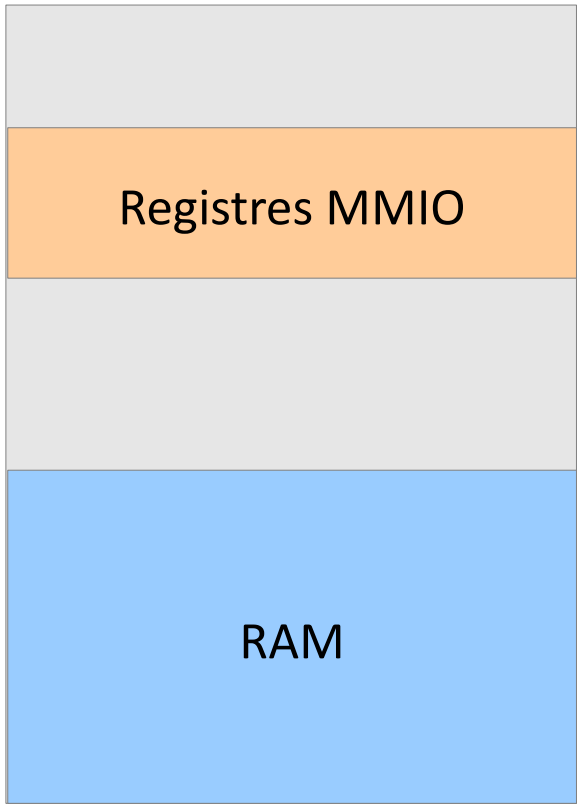
Florent Gluck – Florent.Gluck@hesge.ch

Version 0.3

h e p i a
Haute école du paysage, d'ingénierie
et d'architecture de Genève

Hes·so GENÈVE
Haute École Spécialisée
de Suisse occidentale

Espace d'adresses



Espace d'adresses accédé avec des instructions d'accès classiques (`MOV`)

Espace d'adresses séparé, uniquement accessible avec des instructions spéciales (`IN`/`OUT` sur x86)

Port I/O vs Memory-Mapped I/O

- Entrées/sorties mappées en mémoire (Memory-mapped I/O - MMIO)
  - Même bus d'adresse pour adresser mémoire et périphériques mappés en mémoire
  - Accès aux périphériques avec instructions classiques (instr. `MOV`)
  - Méthode la plus utilisée sur les différentes arch. processeurs
- Ports d'entrées/sorties (Port I/O - PIO)
  - Mémoire et périphériques utilisent des espaces d'adresses différents
  - CPU utilise des instructions spéciales pour accéder aux périphériques
    - Sur x86 : instructions `IN` et `OUT`

x86 mémoire basse (< 1 MB)

START	END	SIZE	TYPE	DESCRIPTION
Low Memory (the first MiB)				
0x00000000	0x000003FF	1 KB	RAM - partially unusable	Real Mode IVT (Interrupt Vector Table)
0x00000400	0x000004FF	256 bytes	RAM - partially unusable	BDA (BIOS data area)
0x00000500	0x00007BFF	almost 30 KB	RAM (free for use)	Conventional memory
0x00007C00	0x00007DFF	512 bytes	RAM - partially unusable	Your OS BootSector
0x00007E00	0x0007FFFF	480.5 KB	RAM (free for use)	Conventional memory
0x00080000	0x0009FBFF	~120 KB, depending on EBDA size	RAM (free for use)	Conventional memory
0x0009FC00	0x0009FFFF	1 KB	RAM (unusable)	EBDA (Extended BIOS Data Area)
0x000A0000	0x000FFFFF	384 KB	various (unusable)	Video memory, ROM Area

Ports sur x86

- Sur x86, accès aux ports avec les instructions `IN` et `OUT`
- Instruction `OUT` permet d'écrire à une adresse
- Instruction `IN` permet de lire la valeur à une adresse
- Adresse du port **toujours** spécifiée dans registre `DX`
- Lecture ou écriture se fait toujours avec `AX`/`AL` selon la taille à lire ou écrire (16-bits ou 8-bits)
- Examples:

Ecrit le byte 4 dans le port 0x60 :

```
mov     dx,0x60
mov     al,4
out     dx,al
```

Lit 16-bits depuis le port 0x60 :

```
mov   dx,0x60
in    ax,dx
```

VGA Text Mode

- Toute carte graphique PC offre un mode texte de 80 colonnes par 25 lignes (“mode texte VGA”)
- 16 couleurs disponibles, cf. Table 1
- Le jeu de caractères natif, appelé Code Page 437, est celui de l'IBM PC d'origine :

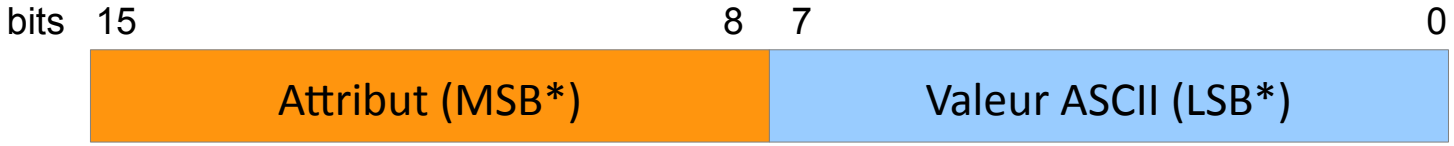
- Table détaillée sur `http://www.ascii-codes.com/`

Value	Color
0	Black
1	Blue
2	Green
3	Cyan
4	Red
5	Magenta
6	Brown
7	Light Gray
8	Dark Gray
9	Light Blue
10	Light Green
11	Light Cyan
12	Light Red
13	Light Magenta
14	Yellow
15	White

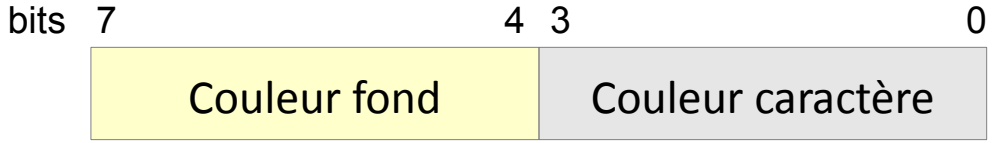
Table 1: couleurs disponibles

Structure du framebuffer

- La mémoire vidéo réservée à l'affichage (*framebuffer*) en mode texte se situe à l'adresses physiques 0xB8000
- Chaque caractère est encodé sur 16-bits, selon :



  - La valeur ASCII est la valeur du caractère à afficher (cf. slide 5)
  - L'attribut est défini comme-suit, où chaque couleur est au choix parmi la Table 1 de la slide précédente :



- La partie visible de la mémoire vidéo est les 4000 premiers bytes (80*25*2) à l'adresse 0xB8000

*MSB signifie Most Significant Byte, LSB Least Significant Byte

Curseur (1)

- Accès curseur via registres du CRTC (Cathode Ray Tube Controller) :
  - Registre de commande : `0x3d4`
  - Registre de données : `0x3d5`
- Mise à jour position curseur :
  - Ecrire commande 0xE dans registre commande
  - Ecrire MSB de position dans registre données
  - Ecrire commande 0xF dans registre commande
  - Ecrire LSB de position dans registre données
- Pour lire position courante curseur ⟶ exactement l'inverse :
  - Ecrire commande 0xE dans registre commande
  - Lire MSB de position dans registre de données
  - Ecrire commande 0xF dans registre de commande
  - Lire LSB de position dans registre de données

Curseur (2)

- Position exprimée en 1D :
  - 0 ⟶ coin supérieur gauche
  - 80*25-1 ⟶ coin inférieur droit

- Curseur peut être désactivé par défaut ; bit 5 du registre “Cursor Start Register” contrôle état curseur ⟶ 0 pour activer, 1 pour désactiver.

  Pour activer curseur :
  - Ecrire commande 0xA dans registre de commande
  - Mettre bit 5 à 0 dans registre de données

- Les 5 premiers bits des registres “Cursor Start Register” (0xA) et “Cursor End Register” (0xB) permettent changer apparence curseur.
  - Exemple, pour obtenir curseur plein :
    - Mettre 5 premiers bits du registre “Cursor Start Register” à 0
    - Mettre 5 premiers bits du registre “Cursor End Register” à 1

Références

- Hardware Level VGA and SVGA Video Programming Information Page
  https://web.stanford.edu/class/cs140/projects/pintos/specs/freevga/vga/vga.htm

- CRT Controller Registers
  https://web.stanford.edu/class/cs140/projects/pintos/specs/freevga/vga/crtcreg.htm