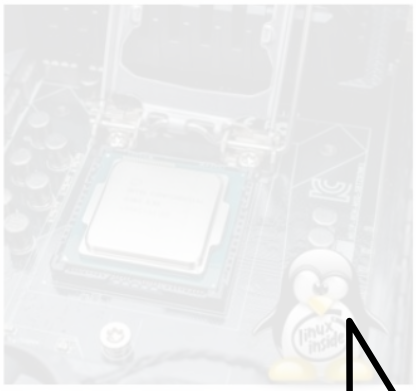
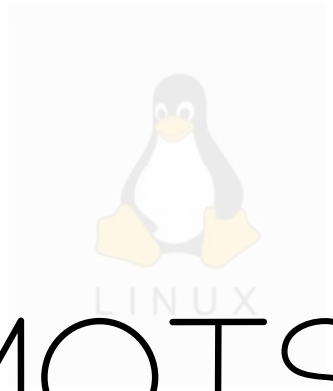
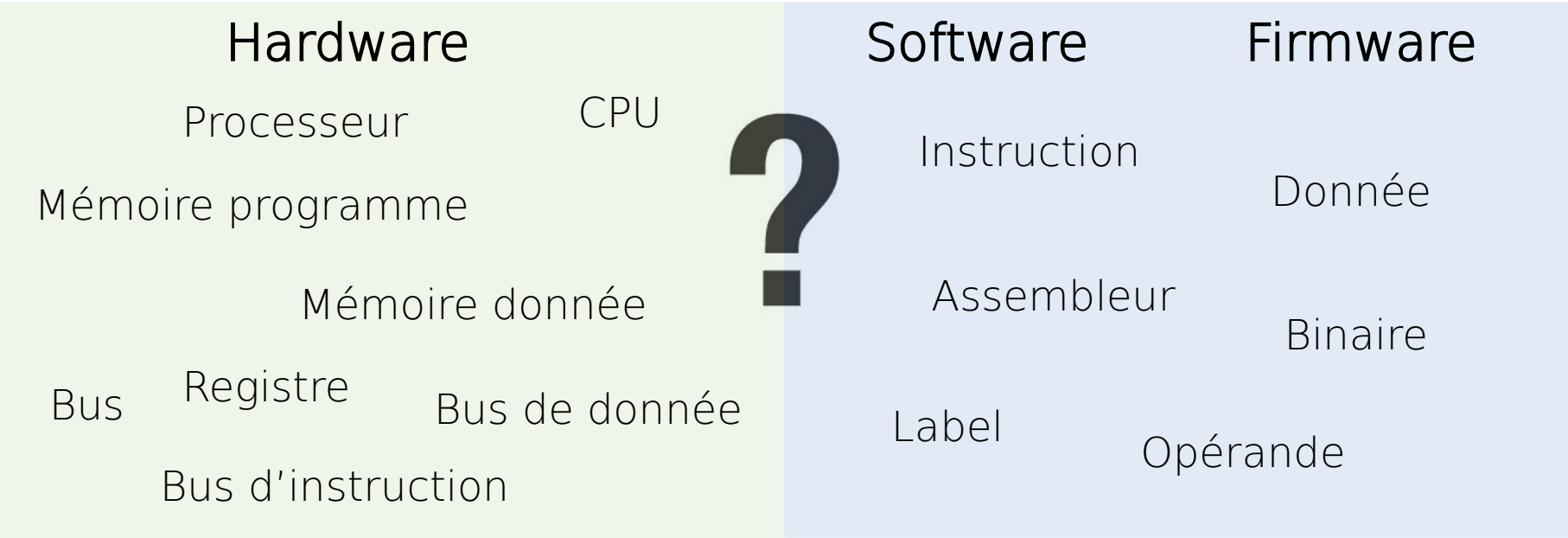



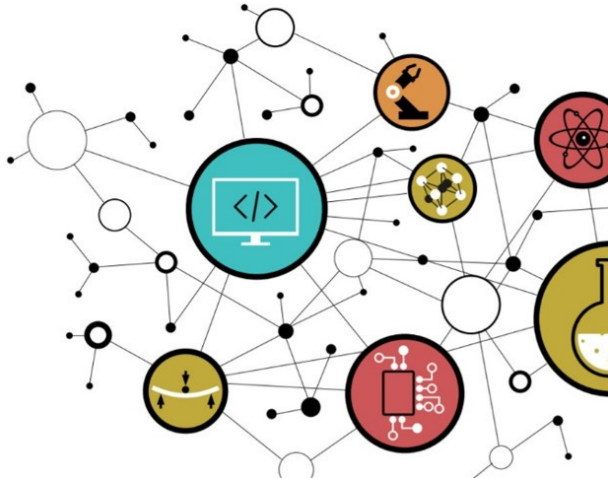
MOTS CLES



Avant d'aborder dans l'enseignement des parties plus technologiques voire standards (toolchain GNU GCC, Kernel Linux, shell Unix, ASM x86_64, carte mère avec chipset Intel, etc), nous allons nous attacher à représenter un jeu de mots clés de base lié au domaine des couches basses des systèmes numériques d'information.



- Langage d'assemblage
- Mémoire et CPU
- Software, Firmware et Hardware
- Périphérique et processeur



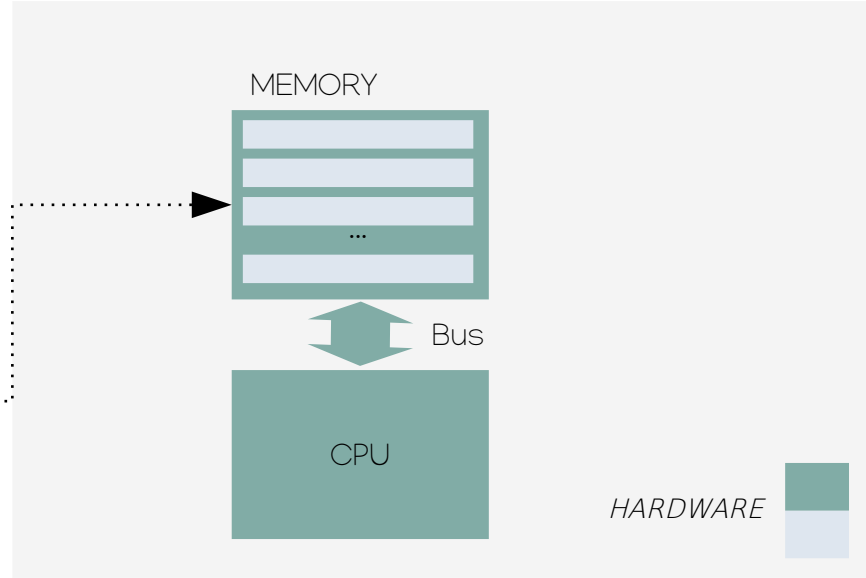
Dans ce document, nous ferons la distinction entre ce qui est matériel (Hardware – bleu) et donc qui peut être touché et tenu par l'homme. De ce qui est la représentation logique de signaux électriques (Software et Firmware – beige). Une machine électronique numérique de traitement de l'information est chargée de stocker, partager et traiter l'information. Les programmes et données sont stockés en mémoire.

For Human

```
MOVE      data1 to data2
ADD       data1 with data2
SUB       data1 with data2
...
```

For Machine

```
01010101
01001000
10001001
   ...
```



SOFTWARE / FIRMWARE

Quelque soit la famille d'un processeur (GPP, MCU, PA, DSP, etc) et sa technologie (RISC-V, Intel, ARM, AMD, PIC18, etc), nous ne trouverons au niveau physique dans la mémoire de la machine que des instructions et des données :

```
MOVE        data1 to data2
ADD         data1 with data2
SUB         data1 with data2
```

- Une instruction est un ordre impératif pour la machine. Un programme est une suite séquentielle d'instructions répondant à un besoin spécifique ou application. Une fois la version d'un programme figée, celui-ci est statique. La séquence d'instructions est inchangée.

- Une donnée représente une information (valeur, caractère, etc). Les données sont en perpétuel changement et mouvement dans la machine. Elles sont manipulées indirectement par les instructions.

L'assembleur (assembly) ou langage d'assemblage est le langage de programmation de plus bas niveau sur la machine. Il est la conversion directe lisible voire éditable par l'homme (texte) du programme exécutable par le CPU de la machine (binaire). Il est de ce fait, le langage le moins universel au monde, car dépendant du jeu d'instructions supporté par le CPU cible (ISA – Instruction Set Architecture). Entre les marchés des ordinateurs et des systèmes embarqués, un grand nombre de fabricants implémentent des modèles d'exécution (RISC ou CISC, Von Neumann ou Harvard, 8-16-32-64bits, entier voire flottant, VLIW ou superscalaire, vectoriel ou scalaire, etc) sur des technologies différentes (x86, x64, ARM, MIPS, C6000, PIC18, etc).

Labels	Instructions	Opérandes
label:	**MOVE**	**register1, register2**
	ADD	**address_data1, register2**
	SUB	**constant, register1**

Instructions présentées à titre illustratif, aucune technologie rattachée

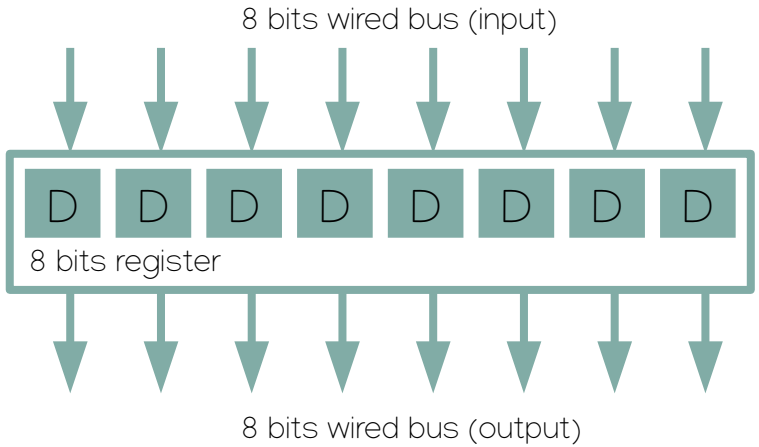
Labels	Instructions	Opérandes
label:	**MOVE**	**register1, register2**
	ADD	**address_data1, register2**
	SUB	**constant, register1**

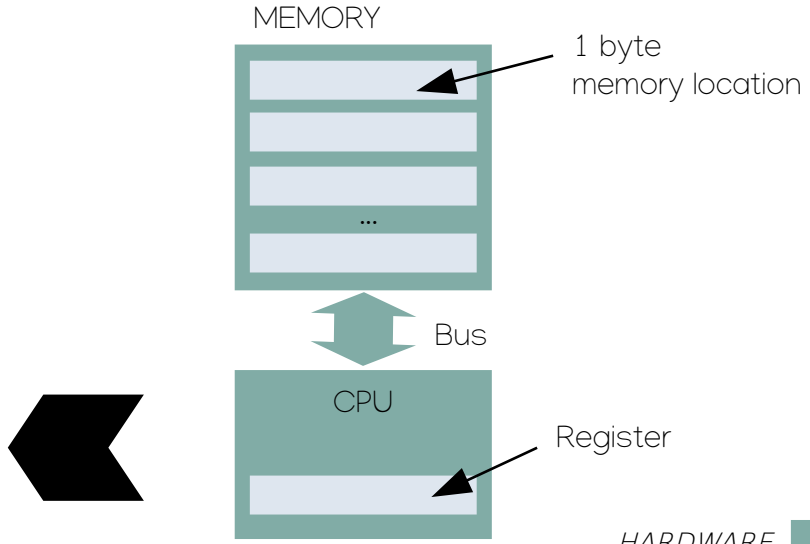
- un label ou étiquette est une référence symbolique (simple chaîne de caractères) représentant l'adresse mémoire logique (emplacement) de la première instruction suivant celui-ci voire d'une variable statique.

- une instruction assembleur est un traitement élémentaire à réaliser par le CPU. Par exemple, charger/load ou sauver/store une donnée depuis ou vers la mémoire, réaliser une opération arithmétique ou logique, déplacer une donnée de registre à registre, etc.

- une opérande, lorsque l'instruction en utilise, est une donnée ou l'emplacement d'une donnée (registre ou adresse mémoire). Nous distinguons l'(es) opérande(s) source(s), de l'opérande de destination pour sauver le résultat (à gauche ou à droite selon convention).

Dans la machine, une donnée ne peut être stockée que dans une case mémoire (technologies SRAM, DRAM, FLASH, etc) ou dans un registre (technologie SRAM – Bascules D). **Un registre** est un ensemble de bascules. Un registre N bits est constitué généralement N bascules D. De même, les données circulent dans la machine sur des bus de communication filaires. **Un bus** est un ensemble de conducteurs électriques physiques.





http://ressources.univ-lemans.fr/AccesLibre/UM/Pedago/physique/02/electro/registres.html

Observons une implémentation technologique : Programme source en langage C, ASM x86_64 en syntaxe AT&T généré par GCC sous Linux

Programme source C

```
char inc(char bar);

int main(void)
{
char foo;

    foo = inc(1);

return 0;
}

char inc(char bar)
{
return bar+1;
}
```

Programme ASM traduit depuis le C

```
Labels  Instructions  Opérandes
main:   push    %rbp
        mov     %rsp,%rbp
        sub     $0x10,%rsp
        mov     $0x1,%edi
        call    4004f2 <inc>
        mov     %al,-0x1(%rbp)
        mov     $0x0,%eax
        leave
        ret
inc:    push    %rbp
        mov     %rsp,%rbp
        mov     %edi,%eax
        mov     %al,-0x4(%rbp)
        movzbl  -0x4(%rbp),%eax
        add     $0x1,%eax
        pop     %rbp
        ret
```

Programme binaire converti depuis l'ASM

```
Instructions binaires   Adresses
55                      4004d6 <main>:
48 89 e5                4004d7:
48 83 ec 10             4004da:
bf 01 00 00 00          4004de:
e8 0a 00 00 00          4004e3:
88 45 ff                4004e8:
b8 00 00 00 00          4004eb:
c9                      4004f0:
c3                      4004f1:
55                      4004f2 <inc> :
48 89 e5                4004f3:
89 f8                   4004f6:
88 45 fc                4004f8:
0f b6 45 fc             4004fb:
83 c0 01                4004ff:
5d                      400502:
c3                      400503:
```

SOFTWARE
ou
Logiciel

```
char inc(char bar);

int main(void)
{
char foo;

    foo = inc(1);

return 0;
}

char inc(char bar)
{
return bar+1;
}
```

```
main:   push   %rbp
        mov    %rsp,%rbp
        sub    $0x10,%rsp
        mov    $0x1,%edi
        call   4004f2 <inc>
        mov    %al,-0x1(%rbp)
        mov    $0x0,%eax
        leave
        ret
inc:    push    %rbp
        mov     %rsp,%rbp
        mov     %edi,%eax
        mov     %al,-0x4(%rbp)
        movzbl  -0x4(%rbp),%eax
        add     $0x1,%eax
        pop     %rbp
        ret
```

Programme C et ASM dans fichier texte (.c, .h, .s, .asm, etc) pour l'humain (développeur)

FIRMWARE
ou
Micrologiciel

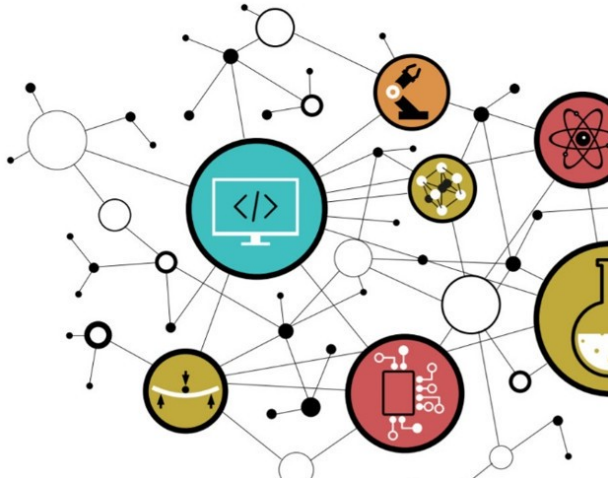
```
55
48 89 e5
48 83 ec 10
bf 01 00 00 00
e8 0a 00 00 00
88 45 ff
b8 00 00 00 00
c9
c3
55
48 89 e5
89 f8
88 45 fc
0f b6 45 fc
83 c0 01
5d
c3
```

Programme binaire dans fichier binaire (ELF, COFF,etc) pour la machine

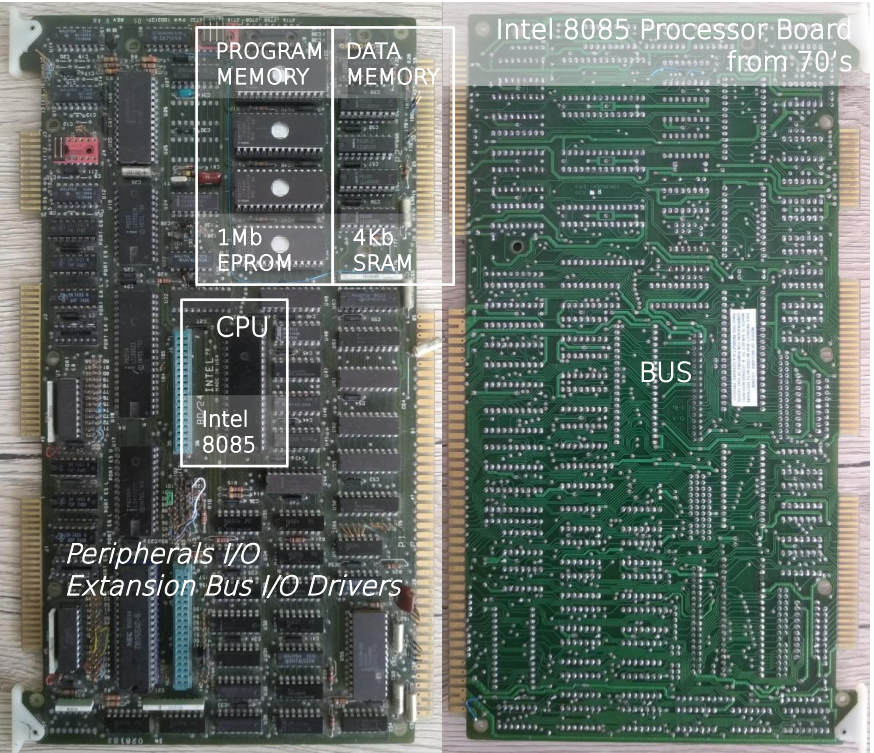
MEMORY MAP
ou
Mapping mémoire

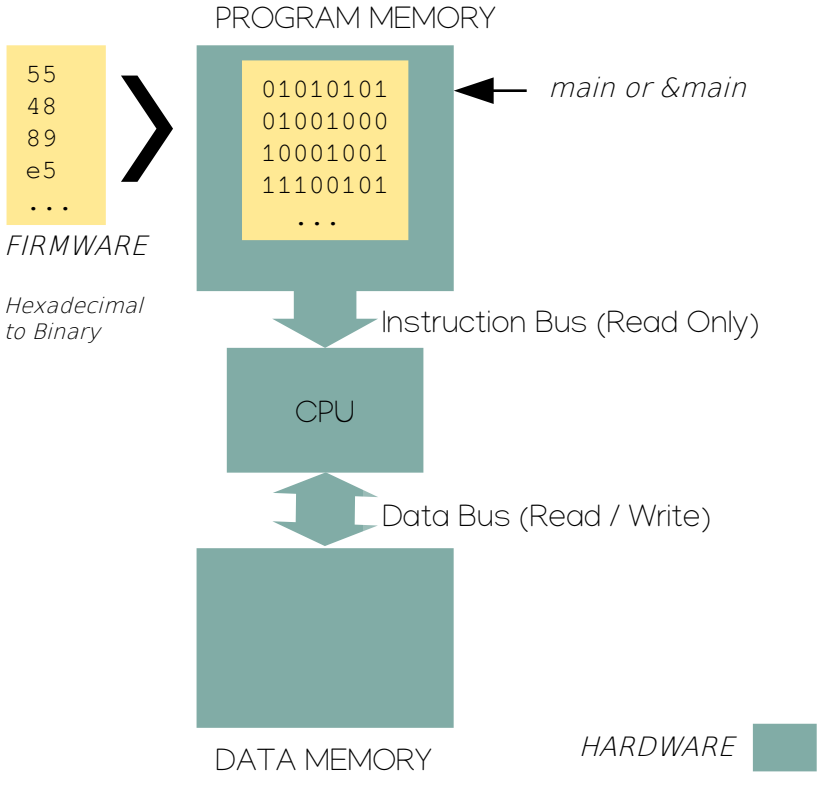
```
4004d6 <main>:
4004d7:
4004da:
4004de:
4004e3:
4004e8:
4004eb:
4004f0:
4004f1:
4004f2 <inc>:
4004f3:
4004f6:
4004f8:
4004fb:
4004ff:
400502:
400503:
```

- Langage d'assemblage
- **Mémoire et CPU**
- Software, Firmware et Hardware
- Périphérique et processeur

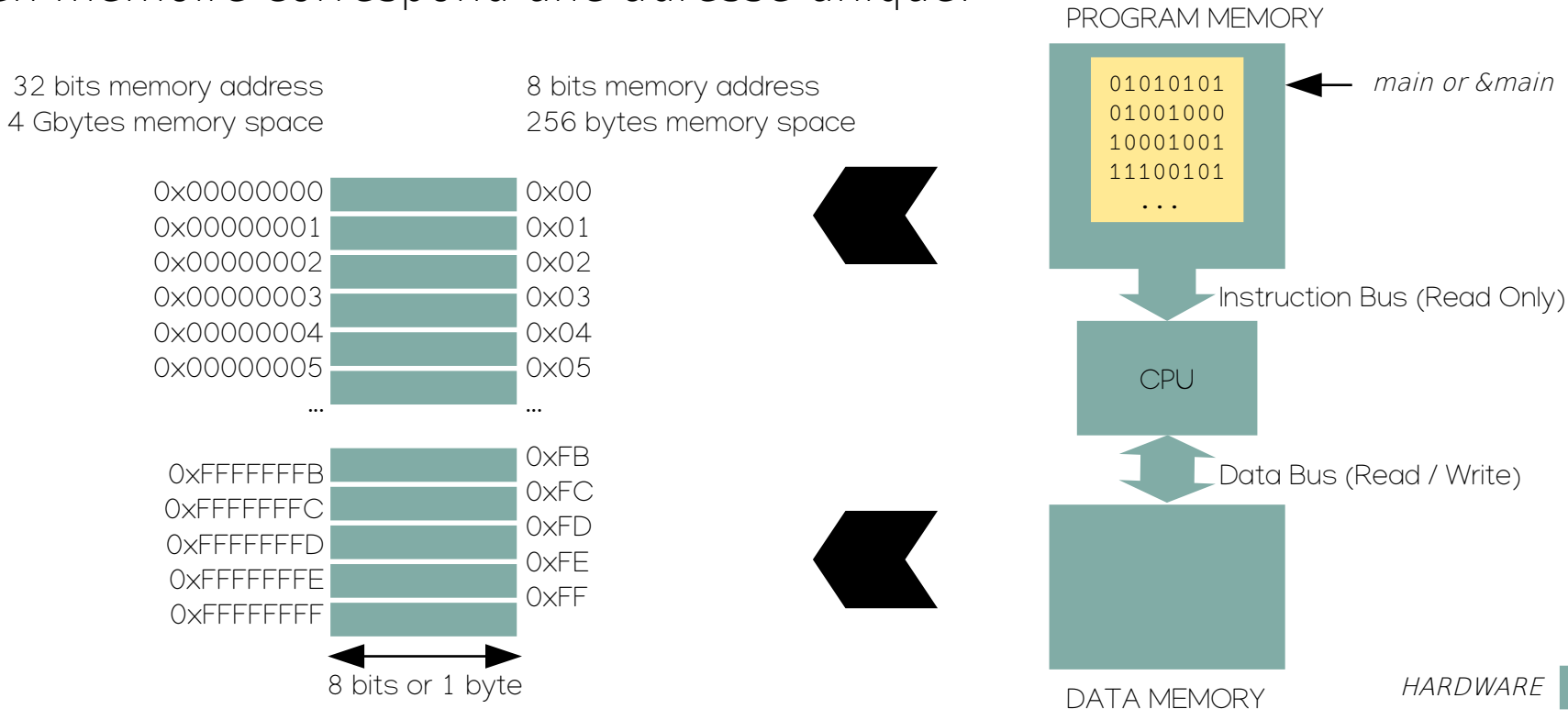


Sur la majorité des processeurs actuels, le stockage des programmes binaires et des données se font dans des mémoires et technologies physiquement séparées autour du CPU.

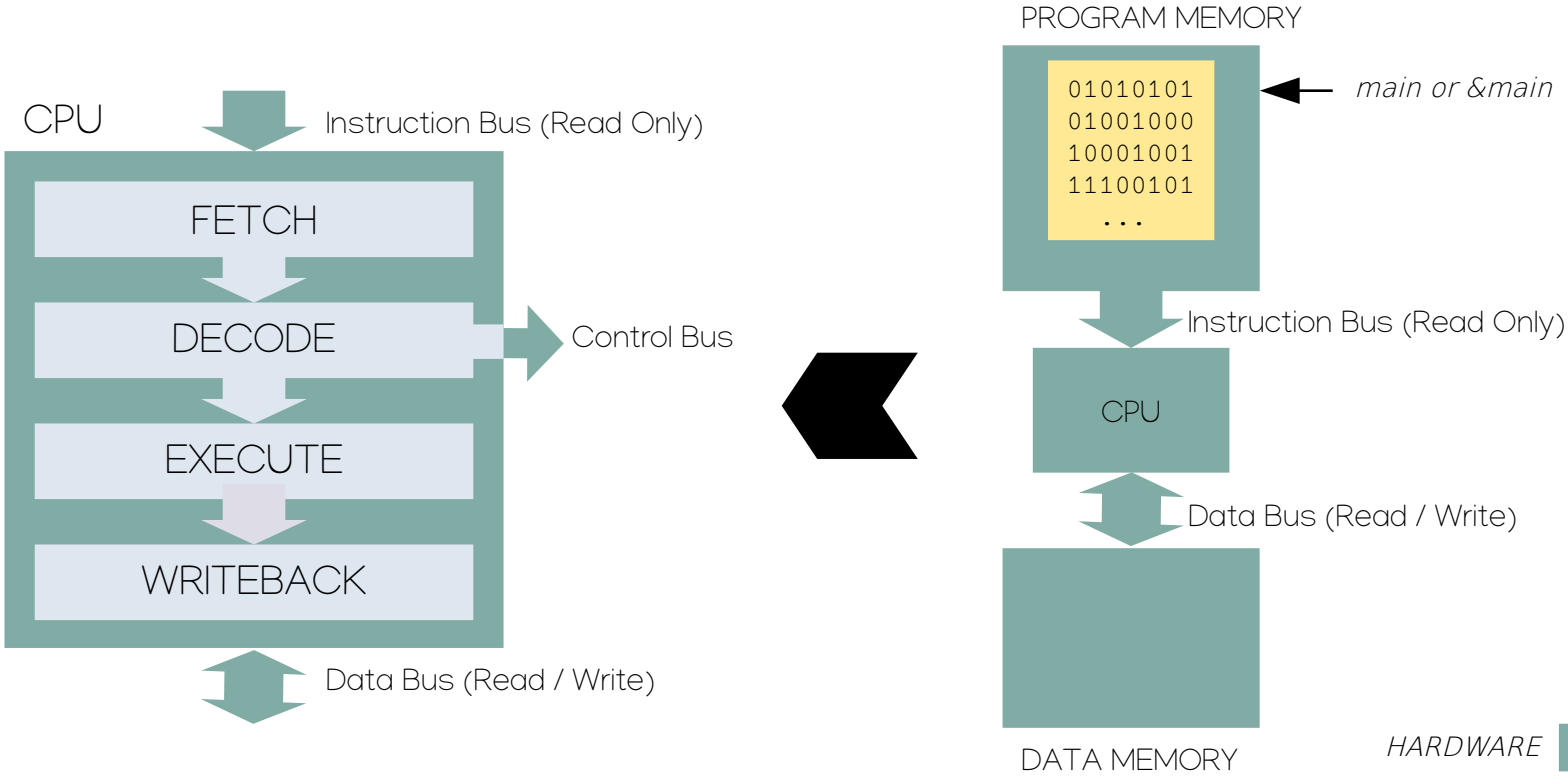




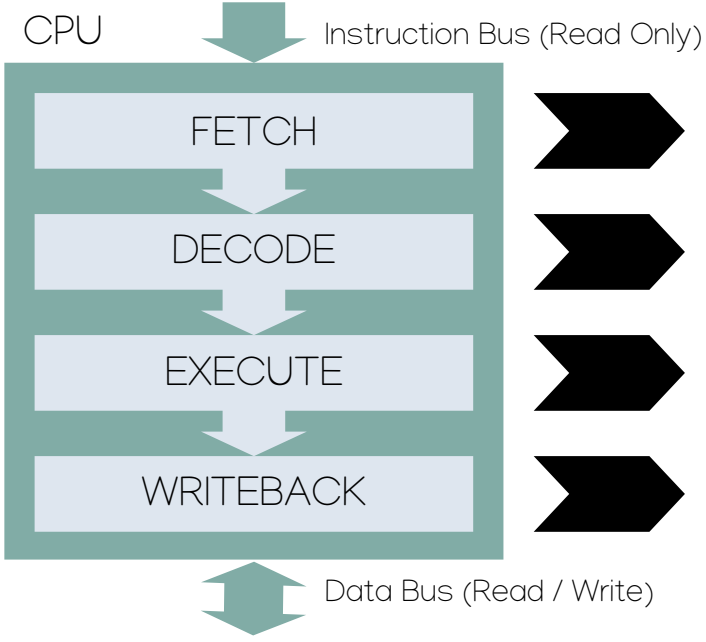
Les modèles mémoires vus du CPU (hors mémoire cache et mémoire de masse) sont dits **adressables par octet**. A chaque octet de stockage en mémoire correspond une adresse unique.



Le CPU (Central Processing Unit) est l'unité de contrôle du processeur dans son ensemble. Le CPU ordonne à tous les composants de la machine. Il est le maître des bus.



Un CPU implémente une machine d'états matérielle. Sauf en veille, le CPU exécutera sans arrêt la même suite séquentielle de traitements (Fetch, Decode, Execute et Writeback). Chaque étape se fait en parallèle des autres sur un même rythme. Nous parlons de **Pipeline hardware**.



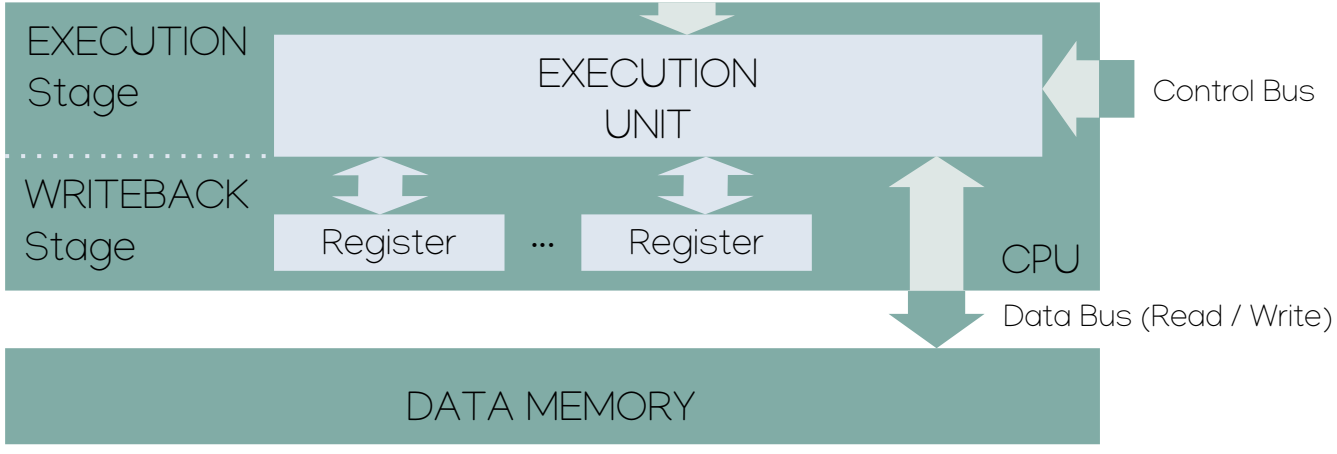
COPIER une instruction binaire à exécuter de la mémoire programme vers le registre de fetch du CPU

IDENTIFIER l'instruction binaire et ORDONNER par le bus de contrôle aux composants de la machine

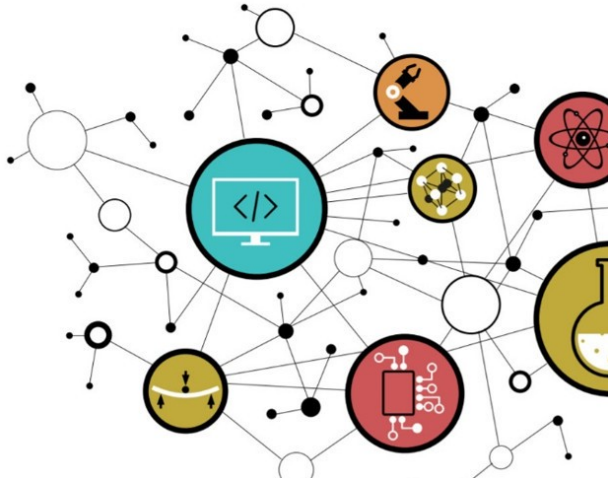
FAIRE le traitement (opération Arithmétique et Logique, déplacement de donnée, calcul d'adresse, etc)

SAUVER " éventuellement " le résultat d'un traitement dans un registre CPU ou en mémoire donnée

L'étage d'exécution est constitué d'une voire plusieurs unités matérielles d'exécution (EU ou Execution Unit). L'unité d'exécution est capable de réaliser des opérations de calcul arithmétique et logique (+, -, *, etc) sur différents formats de donnée selon la technologie CPU (entier 8-16-32-64bits voire flottants). L'unité de calcul est parfois nommée ALU (Arithmetic Logic Unit). L'unité d'exécution est également capable de manipuler (chercher et stocker) une donnée soit dans un registre CPU pour une utilisation en cours soit dans la mémoire donnée.



- Langage d'assemblage
- Mémoire et CPU
- Software, Firmware et Hardware
- Périphérique et processeur



SOFTWARE

FIRMWARE

```
int data1 = 1, data2 ;

data2 = data1 ;
data1 = data1 + data2 ;
data1 = data1 - data2 ;
```

```
MOVE    data1 to data2
ADD     data1 with data2
SUB     data1 with data2
...
```

```
00110011
11000010
11010010
...
```

- Un software ou logiciel (hérité du mot logique) représente une séquence d'instructions abstraites au regard du fonctionnement physique et électronique de la machine (signaux électriques à logique binaire). Les programmes logiciel sont rangés dans des fichiers texte (.c, .h, .s, .cpp, etc pour l'humain) dépendant de la technologie du langage de programmation utilisé (C, ASM, C++, D, JAVA, etc). Ces fichiers et programmes sont à visée de l'homme (développeur ou administrateur). Un logiciel peut être qualifié de système, applicatif, standard, spécifique, libre, propriétaire, etc

SOFTWARE

FIRMWARE

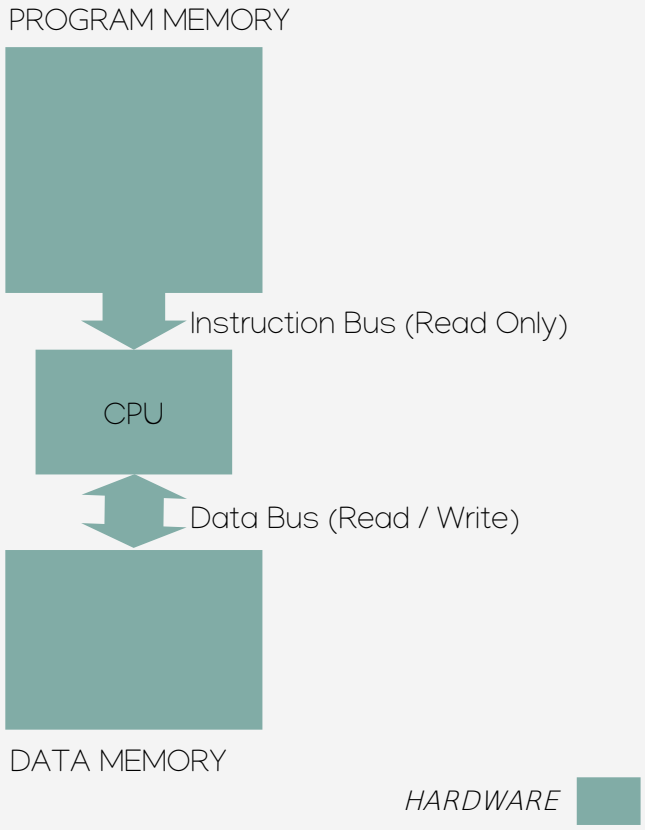
```
int data1 = 1, data2 ;

data2 = data1 ;
data1 = data1 + data2 ;
data1 = data1 - data2 ;
```

```
MOVE    data1 to data2
ADD     data1 with data2
SUB     data1 with data2
...
```

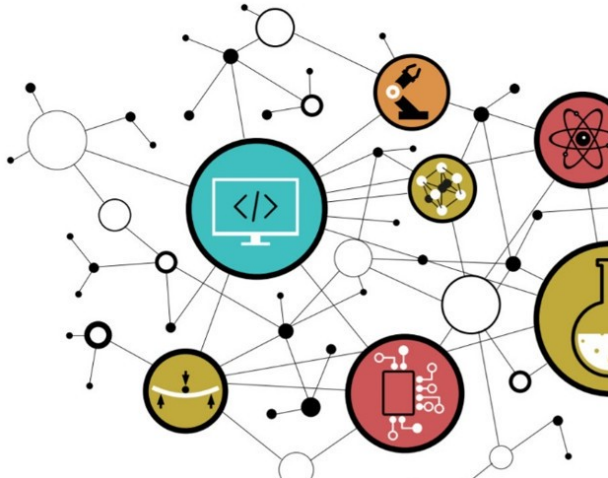
```
00110011
11000010
11010010
...
```

- Un firmware ou micrologiciel représentera dans cet enseignement la conversion directe sans interprétation de la séquence d'instructions assembleur ainsi que l'ensemble des données statiques générées ou allouées à la compilation et l'édition des liens. Un langage d'assemblage n'étant ni standard ni normé mais étant spécifique à l'architecture CPU cible, le code binaire correspondant est également spécifique. Un CPU Intel x86_64 sur ordinateur ne pourra pas exécuter un code binaire pour CPU ARM pour smartphone. En fonction du contexte, nous pourrons appeler quelquefois le code binaire seul le firmware (sans les données et sections de données statiques)

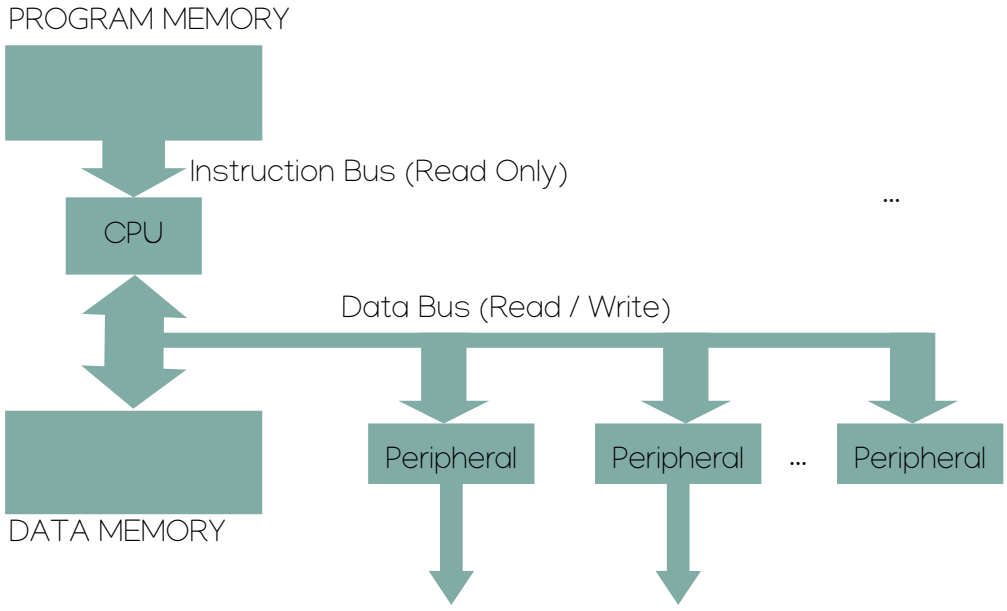
- L'architecture hardware ou matérielle correspond au système électronique numérique physique de traitement de l'information. Il s'agit d'un système à logique binaire (état logique 1 ou 0, haut ou bas). Ces systèmes proposent 3 services :
    - STOCKER l'information : Mémoires programme et donnée
    - TRAITER l'information : CPU
    - PARTAGER l'information : Bus et périphériques



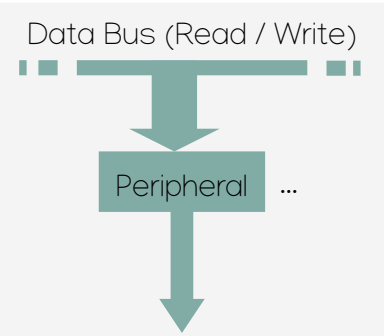
- Langage d'assemblage
- Mémoire et CPU
- Software, Firmware et Hardware
- **Périphérique et processeur**

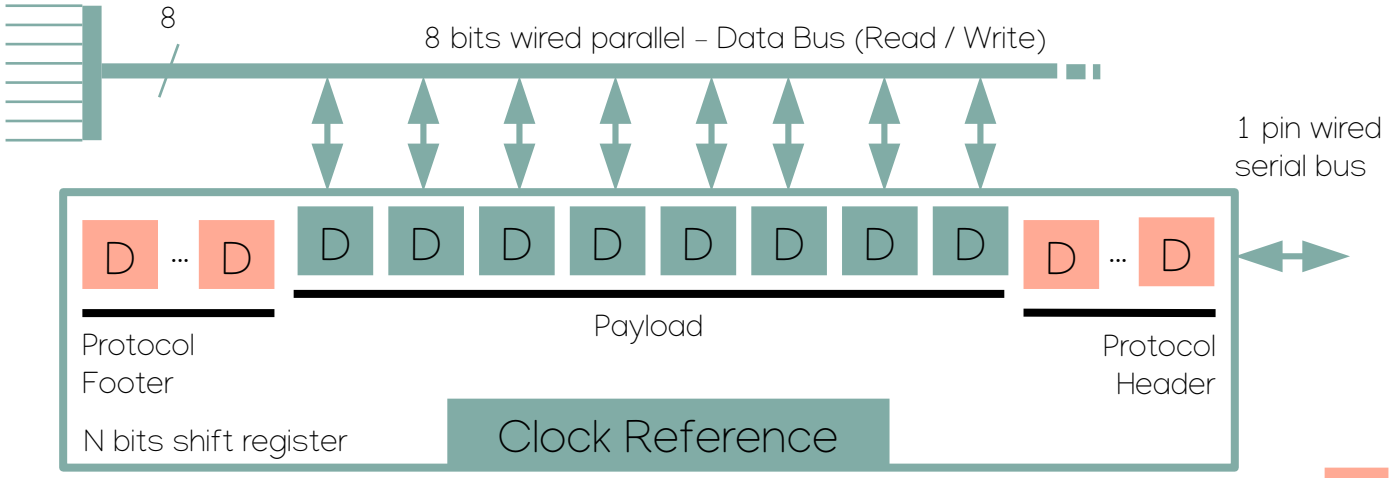


L'échange d'information avec l'extérieur de la machine se fait en passant par les **fonctions matérielles périphériques**, plus communément appelés périphériques. Nous parlons de périphériques par rapport au couple CPU/Mémoire permettant de stocker et traiter l'information. Chaque périphérique implémente une fonction matérielle spécifique.



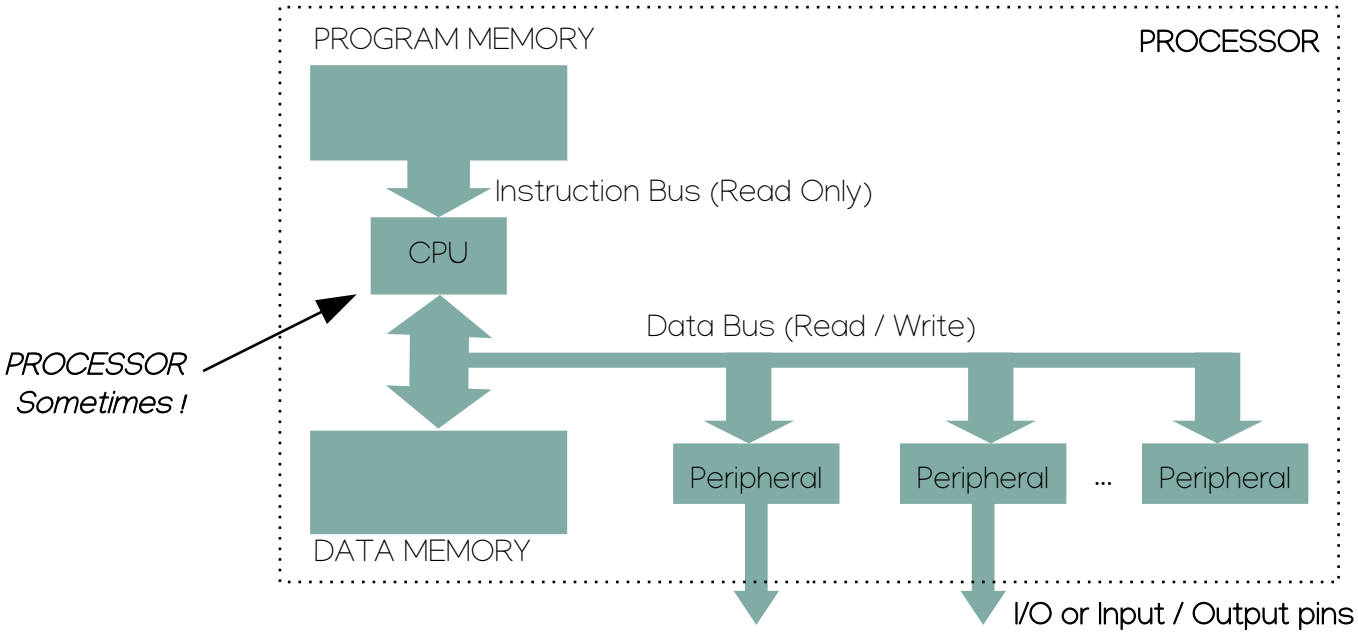
Beaucoup de fonctions périphériques sont chargées de partager l'information avec l'extérieur de la machine. Chaque périphérique implémente alors un protocole de communication particulier (USB, Ethernet, UART, SPI, I2C, etc). Les données utiles (payload) circulent par le bus parallèle de donnée du processeur pour être chargées dans les registres à décalage des périphériques de communication. Les données sont transmises ensuite bit après bit en série à un rythme donné.



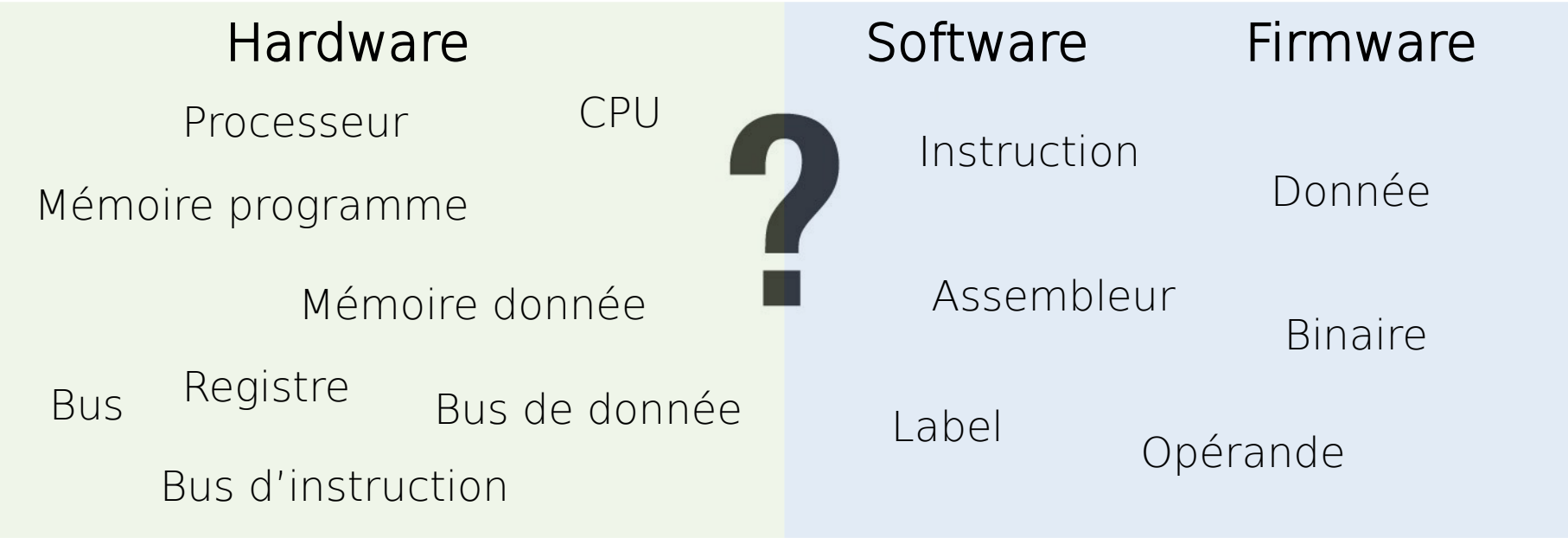


http://ressources.univ-lemans.fr/AccesLibre/UM/Pedago/physique/02/electro/registres.html

Dans cet enseignement, en fonction du contexte, nous nommerons **processeur** soient les familles de machines numériques matérielles implémentant un ensemble CPU/Mémoires/Bus/Périphériques (MCU, AP, DSP, MMPA, GPU, etc) soit dans certains cas le CPU seul (GPP, etc).



Ce jeu de mots clés est central afin de bien se comprendre dans la suite des enseignements en Systèmes Embarqués en formation. Progressivement, la suite de l'enseignement nous permettra d'apprécier certaines subtilités d'implémentation technologiques logicielles et matérielles de solutions leaders du marché (GNU\Linux, GCC, Intel, etc)



Merci !