Démonstration 2

IFT6800

18 septembre 2015

Conversion de base

Convertissez en décimal le nombre hexadécimal à 16 bits suivant selon le complément à deux.

$$\texttt{8000}_{16}$$

On observe d'abord que $\texttt{8000}_{16}$ commence par un `1` en binaire, car $\texttt{8}_{16} = \texttt{1000}_2$. Le nombre est donc négatif. Il représente $x < 0$ tel que

$$\begin{aligned}
\texttt{8000}_{16} &= \texttt{FFFF}_{16} - |x| + \texttt{1}_{16} \\
|x| &= \texttt{FFFF}_{16} - \texttt{8000}_{16} + \texttt{1}_{16} \\
|x| &= \texttt{7FFF}_{16} + \texttt{1}_{16} \\
|x| &= \texttt{8000}_{16} \\
&= (8 \times 16^3)_{10} \\
&= 32\,768_{10}
\end{aligned}$$

Ainsi, $\texttt{8000}_{16}$ représente $-32\,768_{10}$ en complément à deux.

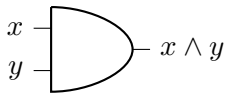
Remarques : 4 digits hexadécimaux interprété avec le complément à deux peuvent contenir les nombres entre $-2^{4\times4-1}$ à $2^{4\times4-1}-1$, inclusivement, i.e. entre 32 768 et 32 767.

Circuits Booléens et arithmétique booléenne

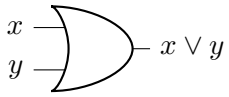
Rappel

On a accès aux portes suivantes pour bâtir des circuits booléens :

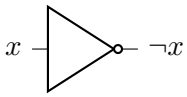
Le ET booléen :



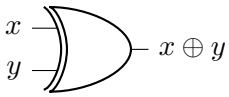
Le OU booléen



Le NON booléen



Le OUX booléen

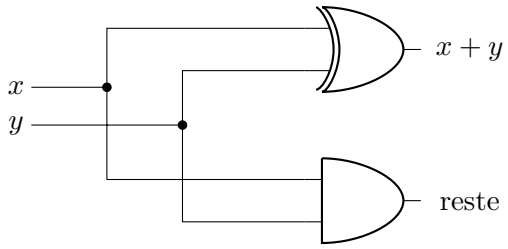


Addition avec retenue

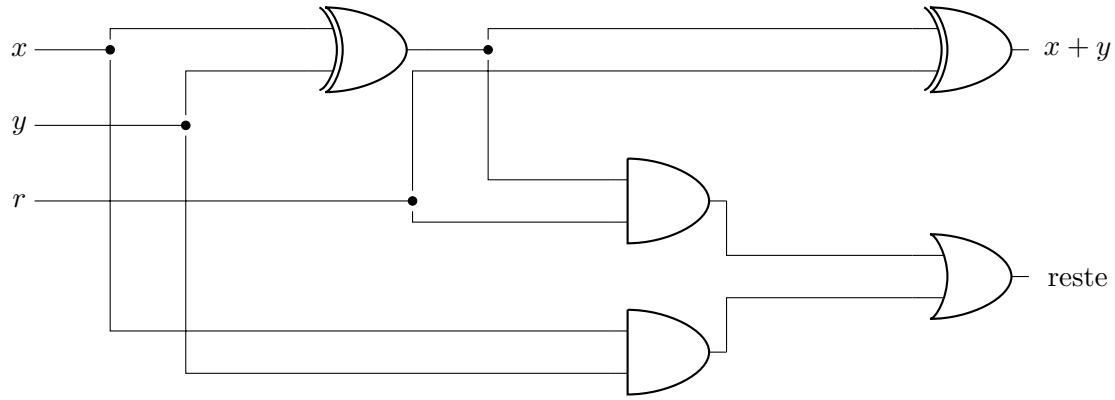
On désire créer un circuit booléen qui sur entrée x, y deux entiers non signés représentés par deux mots binaires de 8 bits, retourne un octet contenant $x + y$.

Attention, si $x + y \geq 2^n$, alors notre addition ne donnera pas la bonne réponse puisqu'il faudrait un bit de plus pour représenter la réponse.

Construisons d'abord un circuit qui sur entrée x, y deux bits retourne l'addition de x et de y dans un bit, et le reste dans un deuxième bit. On appelera circuit : "presqu'addition"



On va maintenant permettre au circuit d'avoir un reste comme entrée supplémentaire. Pour cela, on combine deux presqu'additions



Pour effectuer une addition sur deux registres de 8 bits (contenant respectivement $a_8a_7a_6a_5a_4a_3a_2a_1$ et $b_8b_7b_6b_5b_4b_3b_2b_1$), on effectue une fois le circuit presqu'addition suivi de 7 fois le circuit précédent.

Le 1^e sous-circuit a comme entrée $x = a_1$ et $y = b_1$. Puis, le i^e sous-circuit a comme entrée $x = a_i$ et $y = b_i$ comme entrée r la sortie "reste" du sous-circuit précédent.

Le résultat de l'addition est la concaténation des sorties $x + y$ de chacun des 8 sous-circuits écrits de droite à gauche.

X-TOY

Programme mystère

```
10: 81FF    read R[1]
11: C114    if (R[1] == 0) goto 14
```

```
12: 91FF    write R[1]
13: C010    goto 10
14: 0000    halt
```

Ce programme lit un mot, l'écrit et recommence. Il arrête quand l'usager entre le mot `0000`.

Swap avec variable temporaire

Le programme suivant prend en entrée deux mots qu'il place dans les registres `A` et `B`. Il permute ensuite `A` et `B` à l'aide du registre temporaire `C`.

```
program SwapAvecTemp
// Input:    a, b
// Output:   b, a
// Remarks:  Swaps a et b
// ---------------------------------------------------------------------------

10: 8AFF   read R[A]
11: 8BFF   read R[B]
12: 1C0A   R[C] <- R[A]
13: 1A0B   R[A] <- R[B]
14: 1B0C   R[B] <- R[C]
15: 9AFF   write R[A]
16: 9BFF   write R[B]
17: 0000   halt
```

Assignation de 16 bits - Avec shift

Le programme assigne au registre `A` le mot `DBA1` et l'imprime. Pour ce faire, il entre `DB` dans `A`, déplace son contenu de 8 bits vers la gauche et lui ajoute le contenu de `B` initialisé à `A1`.

```
program Assigner16bits
// Input:    Rien
// Output:   DBA1
// Remarks:  Assigne DBA1 a une variable et l'imprime
// ---------------------------------------------------------------------------

10: 7ADB    R[A] <- DB
11: 9AFF    write [A]
```

```
12: 7808    R[8] <- 08
13: 5AA8    R[A] <- R[A] << R[8]
14: 9AFF    write [A]
15: 7BA1    R[B] <- A1
16: 4AAB    R[A] <- R[A] ^ R[B]
17: 9AFF    write [A]
18: 0000    halt
```

Assignation de 16 bits - Avec accès à la mémoire

Le programme assigne au registre `A` le mot `DBA1` et l'imprime. Pour ce faire, il initialise l'instruction `13` à `DB` qu'il copie dans la variable `A`. L'instruction à la ligne `13` n'est jamais exécutée.

```
program Assigner16bits
// Input:    Rien
// Output:   DBA1
// Remarks:  Assigne DBA1 a une variable et l'imprime
// ------------------------------------------------------------------------------

10: 8A13    R[A] <- mem[13]
11: 9AFF    write R[A]
12: 0000    halt
13: DBA1
```