

# Exemple d'utilisation du simulateur de processeur Y86

The screenshot shows the Y86 Simulator interface. The top navigation bar includes 'Fichier', 'Édition', 'Affichage', 'Historique', 'Marque-pages', and 'Outils'. The browser address bar shows 'dept-info.labri.fr/ENSEIGNEMENT/archi/js-y86/'. The simulator's main toolbar contains 'Assemble', 'Reset', 'Step', 'Continue', and 'Load'. The interface is divided into several panels:

- SOURCE CODE:** Contains assembly instructions: `1 | pos 0`, `2 | irmovl 268, %eax`, `3 | halt`, `4 |`, `5 |`.
- OBJECT CODE:** An empty panel for the compiled machine code.
- MEMORY:** A table showing memory addresses and values. The values are all 00000000. The stack pointer (ESP) is at 00000000.
- REGISTERS:** A table showing the state of 8 registers: `%eax`, `%ecx`, `%edx`, `%ebx`, `%esp`, `%ebp`, `%esi`, and `%edi`. All registers contain the value 0.
- FLAGS:** A table showing the status of various flags: `SF 0`, `ZF 0`, `OF 0`, `STAT AOK`, `ERR`, and `PC 0x0000`.

Red boxes and arrows highlight specific features: a box around the source code, a box around the memory table, a box around the registers table, and a box around the 'Load' button.

On écrit ici le code source (en langage d'assemblage, par abus de langage on dit aussi en assembleur). On peut charger le programme depuis un fichier texte en cliquant sur l'onglet Load.

Les mots mémoires sont alignés sur 32 bits et utilise la convention little-endian (stockage des octets dans l'ordre inverse, c'est-à-dire en commençant par l'octet de poids faible).

Le Y86 utilise 8 registres de 32 bits

# Exemple d'utilisation du simulateur de processeur Y86

The screenshot shows the Y86 Simulator interface with the following components:

- SOURCE CODE:**

```
1 .pos 0
2 irmovl 268, %eax
3 halt
4
5
```
- OBJECT CODE:**

```
0x0000: .pos 0
0x0000: 30f00c010000 irmovl 268, %eax
0x0006: 00 halt
```
- MEMORY:**

ADDR	VALUE
0000	30f00c01 ◀ EBP ◀ ESP
0004	00000000
0008	00000000
000c	00000000
0010	00000000
0014	00000000
0018	00000000
001c	00000000
0020	00000000
0024	00000000
0028	00000000
002c	00000000
0030	00000000
0034	00000000
0038	00000000
003c	00000000
0040	00000000
0044	00000000
0048	00000000
004c	00000000
0050	00000000
0054	00000000
0058	00000000
005c	00000000
0060	00000000
0064	00000000
0068	00000000
006c	00000000
0070	00000000
0074	00000000
0078	00000000
007c	00000000
0080	00000000
0084	00000000
0088	00000000
008c	00000000
0090	00000000
- REGISTERS:**

Register	Value
%eax	0x00000000 0
%ecx	0x00000000 0
%edx	0x00000000 0
%ebx	0x00000000 0
%esp	0x00000000 0
%ebp	0x00000000 0
%esi	0x00000000 0
%edi	0x00000000 0
- FLAGS:**

Flag	Value
SF	0
ZF	0
OF	0
- STATUS:**

Status	Value
STAT	AOK
ERR	
PC	0x0000

Annotations in red boxes:

- En cliquant sur l'onglet Assemble, on crée le code objet, chaque donnée et instruction est placé en mémoire (little-endian)
- Adresse mémoire (pointe sur des « paquets » de 6 octets)
- Valeur stockée en mémoire au format little-endian
- Adresse mémoire (pointe sur des « paquets » de 4 octets)

# Exemple d'utilisation du simulateur de processeur Y86

The screenshot shows the Y86 Simulator interface. The top menu bar includes 'Fichier', 'Édition', 'Affichage', 'Historique', 'Marque-pages', and 'Outils'. The browser address bar shows 'dept-info.labri.fr/ENSEIGNEMENT/archi/fs-y86/'. The simulator's toolbar contains 'Assemble', 'Reset', 'Step', 'Start', and 'Load' buttons. The main area is divided into four panels: 'SOURCE CODE', 'OBJECT CODE', 'MEMORY', and 'REGISTERS/FLAGS/STATUS'.

**SOURCE CODE:**

```
1 .pos 0
2 irmovl 268, %eax
3 halt
4
5
```

**OBJECT CODE:**

```
0x0000: .pos 0
0x0000: 30f00c010000 | irmovl 268, %eax
0x0006: 00 | halt
```

**MEMORY:**

ADDR	VALUE
0000	30f00c01 ◀ EBP ◀ ESP
0004	00000000
0008	00000000
000c	00000000
0010	00000000
0014	00000000
0018	00000000
001c	00000000
0020	00000000
0024	00000000
0028	00000000
002c	00000000
0030	00000000
0034	00000000
0038	00000000
003c	00000000
0040	00000000
0044	00000000
0048	00000000
004c	00000000
0050	00000000
0054	00000000
0058	00000000
005c	00000000
0060	00000000
0064	00000000
0068	00000000
006c	00000000
0070	00000000
0074	00000000
0078	00000000
007c	00000000
0080	00000000
0084	00000000
0088	00000000
008c	00000000
0090	00000000

**REGISTERS:**

Register	Value
%eax	0x00000000 0
%ecx	0x00000000 0
%edx	0x00000000 0
%ebx	0x00000000 0
%esp	0x00000000 0
%ebp	0x00000000 0
%esi	0x00000000 0
%edi	0x00000000 0

**FLAGS:**

Flag	Value
SF	0
ZF	0
OF	0

**STATUS:**

Status	Value
STAT	AOK
ERR	
PC	0x0000

A red box highlights the 'Step' button in the toolbar, with a red arrow pointing to the instruction 'irmovl 268, %eax' in the object code panel. A text box explains the function of the 'Step' button.

En cliquant sur Step on exécute le code instruction par instruction (en vert l'instruction qui va être exécutée au prochain clic sur Step

# Exemple d'utilisation du simulateur de processeur Y86

The screenshot shows the Y86 Simulator interface. The source code panel on the left contains the following assembly instructions:

```
1 .pos 0
2 irmovl 268, %eax
3 halt
4
5
```

The object code panel in the center shows the compiled instructions:

```
0x0000: | .pos 0
0x0000: 30f00c010000 | irmovl 268, %eax
0x0006: 00 | halt
```

The memory panel on the right shows the memory dump:

ADDR	VALUE
0000	30f00c01 ◀ EBP ◀ ESP
0004	00000000
0008	00000000
000c	00000000
0010	00000000
0014	00000000
0018	00000000
001c	00000000
0020	00000000
0024	00000000
0028	00000000
002c	00000000
0030	00000000
0034	00000000
0038	00000000
003c	00000000
0040	00000000
0044	00000000
0048	00000000
004c	00000000
0050	00000000
0054	00000000
0058	00000000
005c	00000000
0060	00000000
0064	00000000
0068	00000000
006c	00000000
0070	00000000
0074	00000000
0078	00000000
007c	00000000
0080	00000000
0084	00000000
0088	00000000
008c	00000000
0090	00000000

The registers panel at the bottom shows the state of the registers:

REGISTER	VALUE
%eax	0x0000010c 268
%ecx	0x00000000 0
%edx	0x00000000 0
%ebx	0x00000000 0
%esp	0x00000000 0
%ebp	0x00000000 0
%esi	0x00000000 0
%edi	0x00000000 0

The flags panel shows the status of the flags:

FLAG	VALUE
SF	0
ZF	0
OF	0

The status panel shows the current status:

STATUS	VALUE
STAT	AOK
ERR	
PC	0x0006

A red box highlights the first instruction in the source code and the corresponding object code. A red arrow points from the box to the register %eax, which now contains the value 268. Another red arrow points from the box to the memory address 0000, which contains the value 30f00c01.

Après exécution de la 1<sup>ère</sup> instruction, on voit bien que 268 est stocké dans le registre %eax

$(268)_{10} = (10c)_{16}$

en little-endian c01

# Exemple d'utilisation du simulateur de processeur Y86

The screenshot displays the Y86 Simulator interface. The top navigation bar includes 'Fichier', 'Édition', 'Affichage', 'Historique', 'Marque-pages', and 'Outils'. The browser address bar shows 'dept-info.labri.fr/ENSEIGNEMENT/archi/js-y86/'. The simulator's main toolbar contains 'Assemble', 'Reset', 'Step', 'Continue', and 'Load'. The interface is divided into several panels:

- SOURCE CODE:** Line 1: `.pos 0`; Line 2: `irmovl 268, %eax`; Line 3: `halt`; Lines 4 and 5 are empty.
- OBJECT CODE:** Shows the assembly instructions with their addresses: `0x0000: .pos 0`, `0x0000: 30f00c010000 irmovl 268, %eax`, and `0x0006: 00 halt`. A red arrow points from the `halt` instruction in the object code to the `halt` instruction in the source code.
- MEMORY:** A table with columns 'ADDR' and 'VALUE'. The first entry is `0000 30f00c01`, with `EBP` and `ESP` pointers next to it. Other memory locations from `0004` to `0090` all contain `00000000`.
- REGISTERS:** A table showing the state of registers: `%eax 0x0000010c 268`, `%ecx 0x00000000 0`, `%edx 0x00000000 0`, `%ebx 0x00000000 0`, `%esp 0x00000000 0`, `%ebp 0x00000000 0`, `%esi 0x00000000 0`, and `%edi 0x00000000 0`.
- FLAGS:** Shows `SF 0`, `ZF 0`, and `OF 0`. The **STATUS** section shows `STAT HLT`, `ERR`, and `PC 0x0007`.

A red box highlights the text: "Le programme est terminé par l'instruction Halt".