Week1_Exercise1_Corrigé

October 15, 2019

1 Algorithmique et Pensée Computationnelle

[ ]:

2 Exercice 1 - Conversions 10 -> 2

Dans cet exercice, vous devrez convertir des nombres en base 10 vers la base 2. Pour vérifier votre réponse, entrez votre réponse entre les " " de chaque question et cliquez sur `run` en haut de l'écran.

2.0.1 Exercice 1 - a

Convertissez le nombre 10 (base 10) en base 2

[ ]:
```
from Assertion import check_1_a

# -----------------
# VOTRE RÉPONSE ICI
réponse = "1010"
# -----------------

check_1_a(réponse)
```

2.0.2 Exercice 1 - b

Convertissez le nombre 45 (base 10) en base 2

[ ]:
```
from Assertion import check_1_b

# -----------------
# VOTRE RÉPONSE ICI
réponse = "101101"
# -----------------

check_1_b(réponse)
```

2.0.3 Exercice 1 - c

Convertissez le nombre 173 (base 10) en base 2

[ ]:
```
from Assertion import check_1_c

# -----------------
# VOTRE RÉPONSE ICI
réponse = "10101101"
# -----------------

check_1_c(réponse)
```

[ ]:
```

```

Week1_Exercise2_Corrigé

October 15, 2019

1 Algorithmique et Pensée Computationnelle

[ ]:

2 Exercice 2 - Conversions 10 -> [3, 16]

Dans cet exercice, vous devrez convertir des nombres en base 10 vers d'autres bases entre 3 et 16

2.0.1 Exercice 2 - a

Convertissez le nombre 40 (base 10) en base 8

[ ]:
```
from Assertion import check_2_a

# -----------------
# VOTRE RÉPONSE ICI
réponse = "50"
# -----------------

check_2_a(réponse)
```

2.0.2 Exercice 2 - b

Convertissez le nombre 52 (base 10) en base 3

[ ]:
```
from Assertion import check_2_b

# -----------------
# VOTRE RÉPONSE ICI
réponse = "1221"
# -----------------

check_2_b(réponse)
```

2.0.3 Exercice 2 - c

Convertissez le nombre 254 (base 10) en base 16.

N.b Les chiffres au dessus de 10 dans les bases supérieurs sont exprimés en lettres majuscules, e.g 30 -> 1E

[ ]:
```
from Assertion import check_2_c

# -----------------
# VOTRE RÉPONSE ICI
réponse = "FE"
# -----------------

check_2_c(réponse)
```

[ ]:
```

```

Week1_Exercise3

October 15, 2019

1 Algorithmique et Pensée Computationnelle

[ ]:

2 Exercice 3 - Conversions [3, 16] -> 10

Dans cet exercice, vous devrez convertir des nombres en base 3 à 16 vers la base 10

2.0.1 Exercice 3 - a

Convertissez le nombre 10110 (base 2) en base 10

[ ]:
```
from Assertion import check_3_a

# -----------------
# VOTRE RÉPONSE ICI
réponse = "22"
# -----------------

check_3_a(réponse)
```

2.0.2 Exercice 3 - b

Convertissez le nombre 4321 (base 5) en base 10

[ ]:
```
from Assertion import check_3_b

# -----------------
# VOTRE RÉPONSE ICI
réponse = "586"
# -----------------

check_3_b(réponse)
```

2.0.3 Exercice 3 - c

Convertissez le nombre ABC (base 16) en base 10

[ ]:
```
from Assertion import check_3_c

# -----------------
# VOTRE RÉPONSE ICI
réponse = "2748"
# -----------------

check_3_c(réponse)
```

[ ]:
```

```

Week1_Exercise4

October 15, 2019

1 Algorithmique et Pensée Computationnelle

[ ]:

2 Exercice 4 - Additions de nombre binaires

Dans cet exercice, vous devrez additionner des nombres binaires entre eux

2.0.1 Exercice 4 - a

Additionnez `01010101` et `10101010`

[ ]:
```
from Assertion import check_4_a

# -----------------
# VOTRE RÉPONSE ICI
réponse = "11111111"
# -----------------

check_4_a(réponse)
```

2.0.2 Exercice 4 - b

Additionnez `01011111` et `10000001`

[ ]:
```
from Assertion import check_4_b

# -----------------
# VOTRE RÉPONSE ICI
réponse = "11100000"
# -----------------

check_4_b(réponse)
```

2.0.3 Exercice 4 - c

Additionnez `01110100` et `00011010`

[ ]:
```
from Assertion import check_4_c

# -----------------
# VOTRE RÉPONSE ICI
réponse = "10001110"
# -----------------

check_4_c(réponse)
```

[ ]:
```

```

Week1_Exercise5_Corrigé

October 15, 2019

1 Algorithmique et Pensée Computationnelle

[ ]:

2 Exercice 5 - Van Neumann

Dans cet exercice, nous allons simuler une opération d'addition dans le `modèle de Van Neumann`, il va vous être demandé à chaque étape (`FDES`) de donner la valeur des `registres`.

2.1 Etat d'origine

A l'origine, notre `Process Counter` (PC) vaut `00100000`

Dans la mémoire, les instructions sont les suivantes:

Adresse	Valeur
00011111	00100100
00100000	10110110
00100001	11101101

Les registres sont les suivants:

Registre	Valeur
00	11100011
01	01101100
10	00100101
11	00000000

Les opérations disponibles pour notre `unité de contrôle` sont les suivantes:

Numéro	Valeur
00	MOV

Numéro	Valeur
01	XOR
10	ADD
11	SUB

2.2 FETCH

Dans la case suivante, entrez la valeur du `Process Counter` et de l'`Instruction Register` à la **FIN** de l'opération `FETCH`.

[ ]:
```
from Assertion import check_5_FETCH

# -----------------
# PROCESS COUNTER:
PROCESS_COUNTER = "00100001"
# -----------------

# -----------------
# INSTRUCTION REGISTER:
INSTRUCTION_REGISTER = "10110110"
# -----------------

check_5_FETCH(PROCESS_COUNTER, INSTRUCTION_REGISTER)
```

2.3 DECODE

Dans la case suivante, entrez: - **La valeur** de l'opération à exécuter - **L'adresse** du registre dans lequel le résultat doit être enregistré - **La valeur** du premier nombre de l'opération - **La valeur** du deuxième nombre de l'opération

[ ]:
```
from Assertion import check_5_DECODE

# -----------------
# La valeur de l'opération à exécuter:
OPERATION = "ADD"
# -----------------

# -----------------
# L'adresse du registre dans lequel le résultat doit être enregistré:
ADRESSE_RESULTAT = "11"
# -----------------

# -----------------
# La valeur du premier nombre de l'opération
PREMIER_NOMBRE = "01101100"
```

```
# -----------------

# -----------------
# La valeur du deuxième nombre de l'opération
DEUXIEME_NOMBRE = "00100101"
# -----------------

check_5_DECODE(OPERATION, ADRESSE_RESULTAT, PREMIER_NOMBRE, DEUXIEME_NOMBRE)
```

2.4 EXECUTE

Dans la case suivante, entrez le résultat de l'opération

[ ]:
```
from Assertion import check_5_EXECUTE

# -----------------
# La valeur de l'opération à exécuter:
RESULTAT = "10010001"
# -----------------

check_5_EXECUTE(RESULTAT)
```

[ ]:
```

```