

Sprawozdanie

Symulator procesora z użyciem javascript

1. Ogólny wygląd

The image shows a web-based simulator interface for a processor. It features a light blue background. On the left, there is a large, empty white rectangular box, likely for assembly code or comments. To the right of this box is a vertical column of five purple buttons with white text: 'MOV', 'MOVI', 'XCHG', 'EXECUTE', and 'FILLRANDOM'. Below the white box, there is a 4x2 grid of eight rounded rectangular boxes, each representing a processor register. The registers are labeled 'AH', 'AL', 'BH', 'BL', 'CH', 'CL', 'DH', and 'DL'. Each label is in bold black text above a white input field containing the number '0'. The entire interface is contained within a light blue rectangular frame.

Białe pole jest stworzone znacznikiem textarea, przyciski z boku znacznikami button, a obszary AH, AL., itd... są to znaczniki div, w których pola do wprowadzania danych to pola typu input z domyślną wartością 0.

2. Obsługa symulatora

1. Wprowadzenie danych

Dane można wprowadzić do symulatora na 2 sposoby

1)Klikając w pola input w elementach AH, AL., itd... i wprowadzić je ręcznie z użyciem klawiatury.

2) Klikają przycisk **FILL Random**, po czym wszystkie wartości zostaną wpisane do każdego elementu losowo.

Wpisanie ręczne

The screenshot shows the initial state of the assembly simulator. On the left, there is a large empty text box for entering assembly code. Below it, eight registers are displayed in a 4x2 grid: AH (321), AL (0), BH (0), BL (231), CH (231), CL (0), DH (0), and DL (3232). On the right, a vertical column of five purple buttons is visible: MOV, MOVI, XCHG, EXECUTE, and FILL RANDOM.

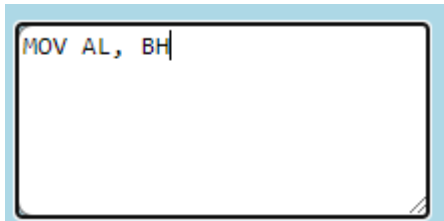
Kliknięcie **FILL RANDOM**

The screenshot shows the simulator after the 'FILL RANDOM' button has been clicked. The registers now contain random values: AH (236), AL (188), BH (101), BL (230), CH (182), CL (55), DH (13), and DL (64). The 'FILL RANDOM' button is highlighted with a green dashed border, and a large green arrow points upwards towards it from the bottom of the screen.

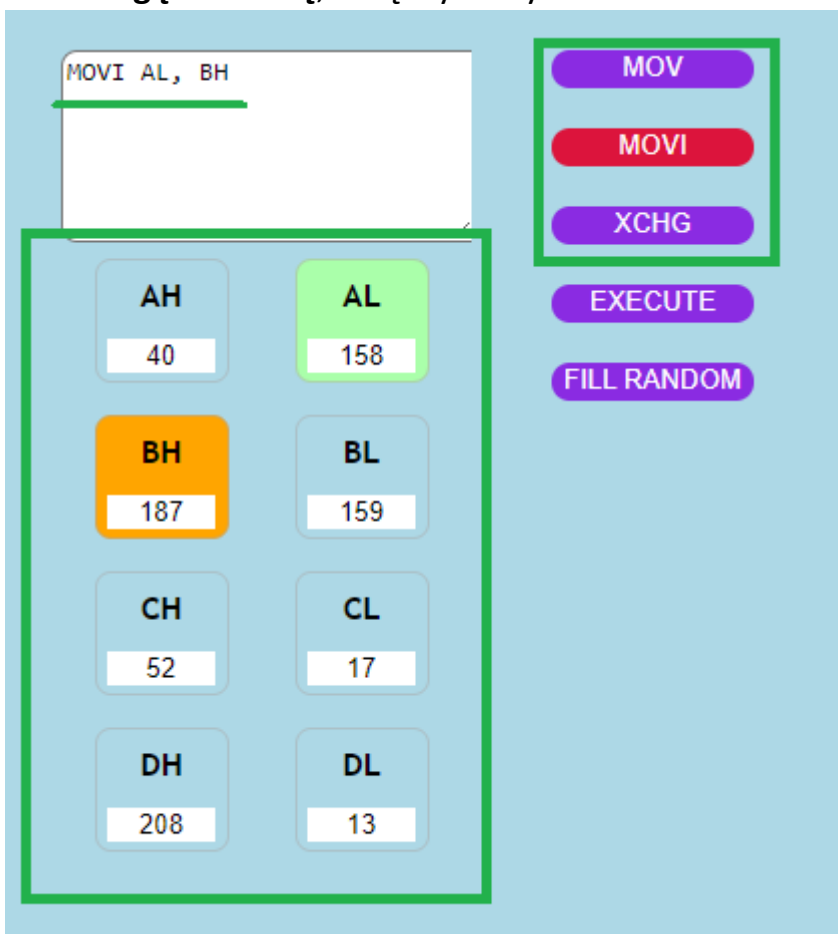
2. Użycie funkcji *MOV*, *MOVI*, *XCHG*

Komendy *MOV*, *MOVI*, *XCHG* są zdefiniowane w kodzie i można je użyć na 2 sposoby, **ale w obu przypadkach należy pamiętać że komendy są pierwszą rzeczą jakie wprowadza się w pole textarea, a dopiero po nich nazwę komórek pomiędzy którymi ma wykonać się wyznaczona funkcja.**

1) W pole textarea, wpisać samodzielnie **nazwę funkcji**, następnie **spacja**, nazwa **pierwszej komórki**, następnie napisać **przecinek** oraz **wcisnąć spację**, na koniec nazwa **drugiej komórki**.



2) Druga opcja wymaga tylko użycia myszki. Najpierw zaznaczamy **funkcję której działanie chcemy użyć**, następnie zaznaczamy **pierwszą oraz drugą komórkę**, między którymi ma zadziałać funkcja.

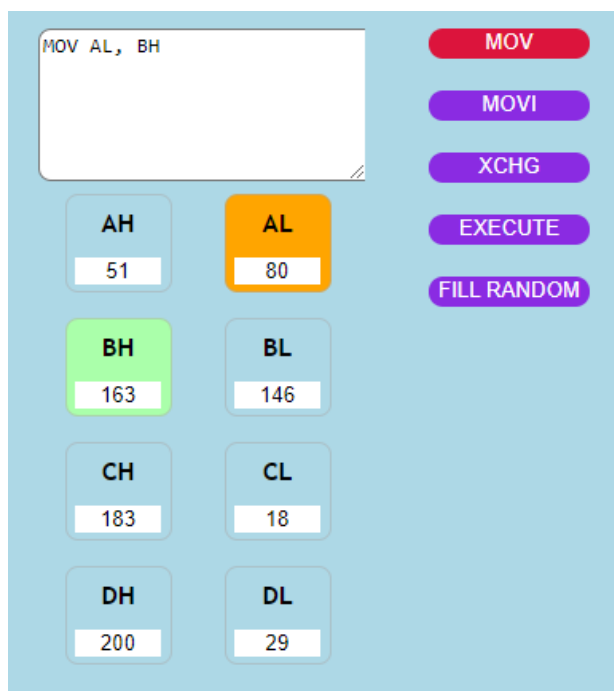


W przypadkach funkcji MOV, MOVI oraz XCHG komórki podświetlają się na kolor pomarańczowy, zielony lub fioletowy .

Kolor pomarańczowy oznacza komórkę, której wartość zostanie zmieniona

Kolor zielony oznacza komórkę, której wartość nie zostanie zmieniona

Kolor fioletowy oznacza elementy, których wartości się zamienią się miejscami



MOV

MOVI AL, BH

AH 0	AL 0
BH 0	BL 0
CH 0	CL 0
DH 0	DL 0

MOV
MOVI
XCHG
EXECUTE
FILL RANDOM

MOVI

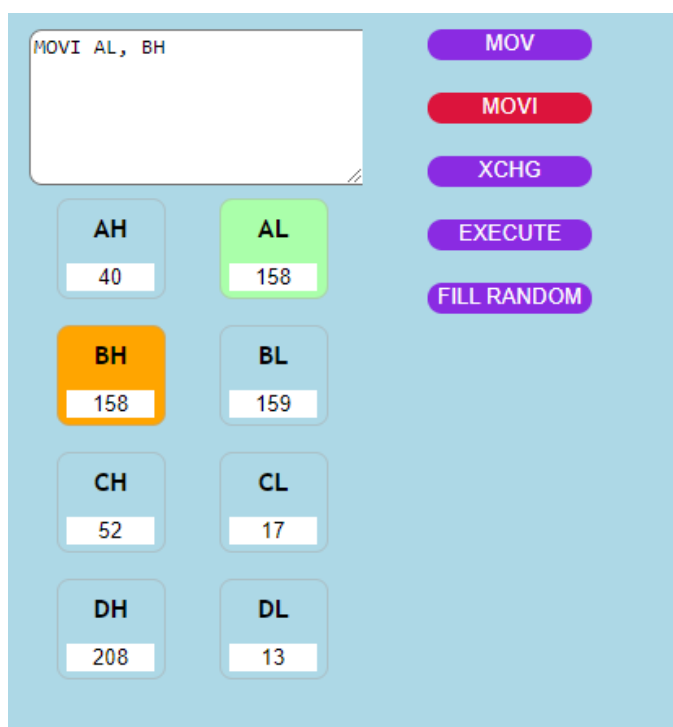
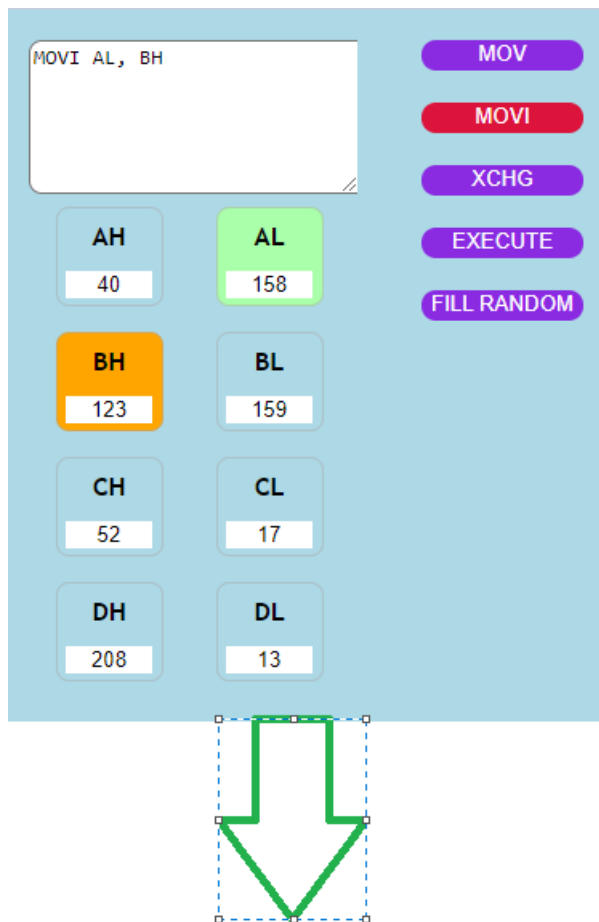
XCHG AL, BH

AH 0	AL 0
BH 0	BL 0
CH 0	CL 0
DH 0	DL 0

MOV
MOVI
XCHG
EXECUTE
FILL RANDOM

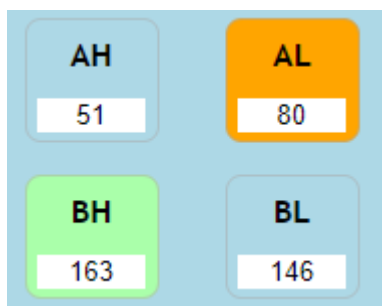
XCHG

3) Ostatnim krokiem jest wywołanie by funkcja wykonała zadany program. Robimy to poprzez naciśnięcie przycisku **EXECUTE**



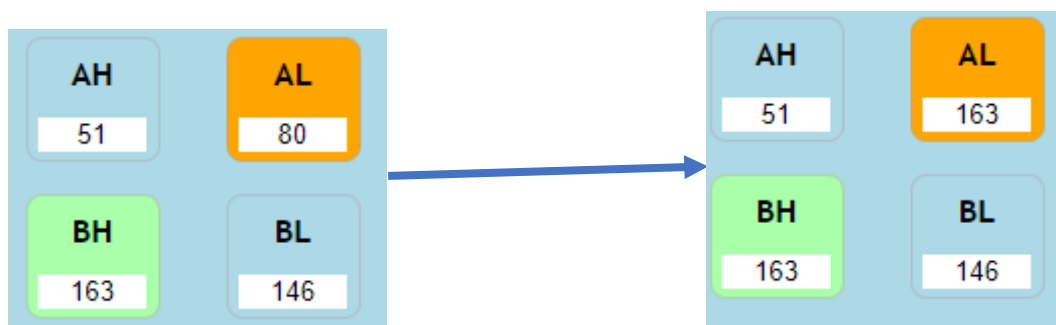
3. Sposób działania poleceń

Działanie będzie pokazane na przykładzie



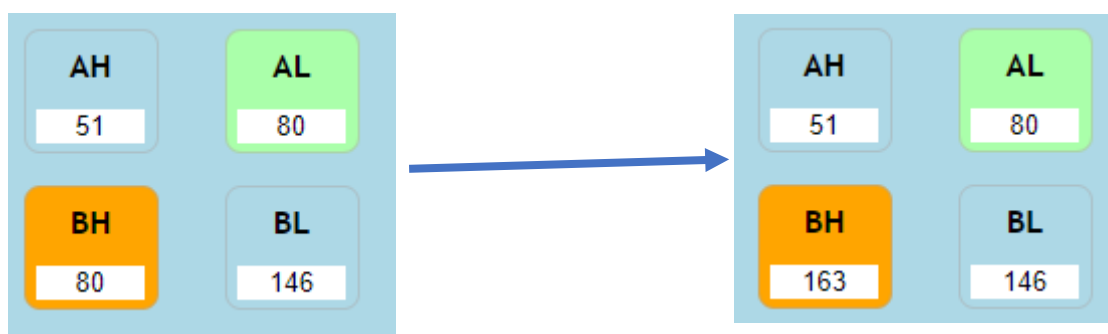
1. MOV

Wartość z drugiego elementu (BL) zostanie przeniesiona do pierwszego elementu (AH)



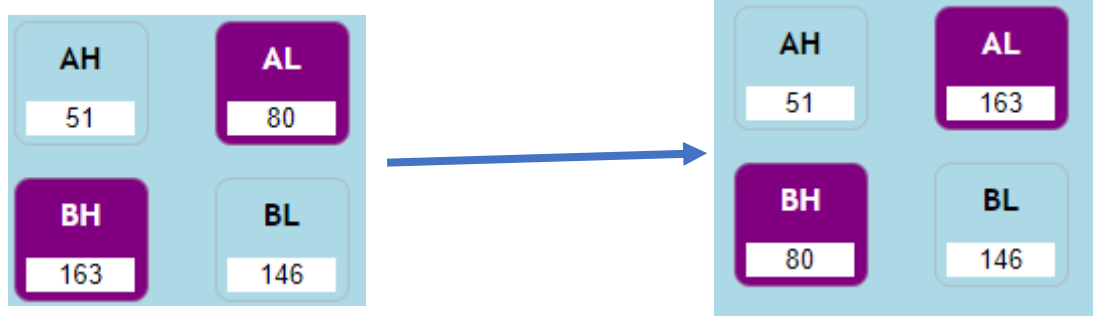
2. MOVI (MOV INVERTED)

Wartość z pierwszego elementu (AH) zostanie przeniesiona do drugiego elementu (BL)



3. XCHG

Wartości zostają zamienione między elementami



Link do repozytorium z kodem

<https://github.com/Gumiin/studia/tree/main/architektura%20systemów>