**SYMULATOR INTEL 8086**

**wYŻSZA SZKOŁA EKONOMII I INFORMATYKI W KRAKOWIE**

**12184 - lab1/1/IES**

**Ciesielski Łukasz**

**github** <https://github.com/LukaszCiesielski-PL/Intel-8086>

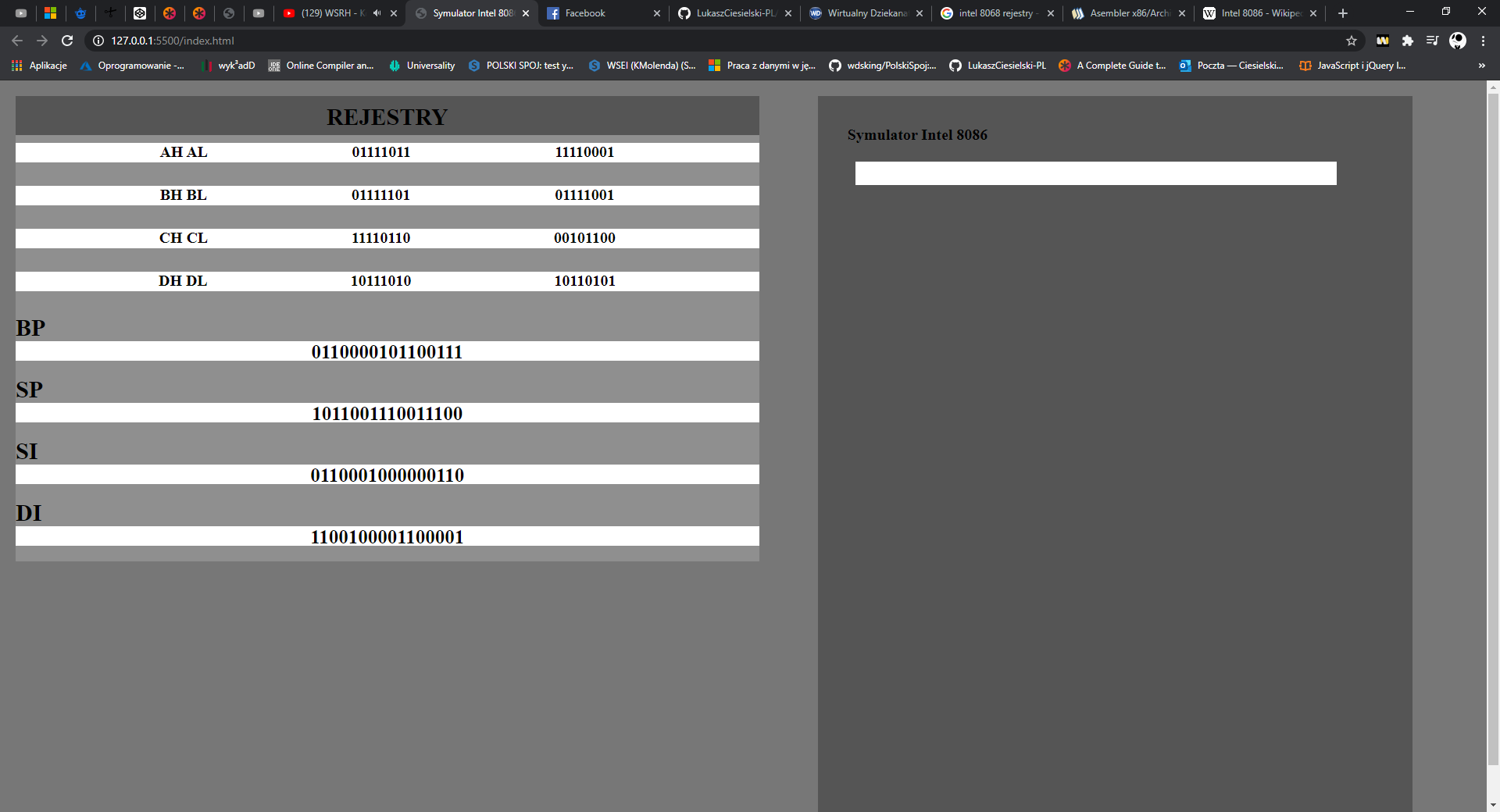
2020

**Budowa i zawartość**

Głównym zadaniem było stworzenie symulatora procesora Intel 8086, którego zadaniem jest symulowanie działania rejestrów i stosów tego procesora.

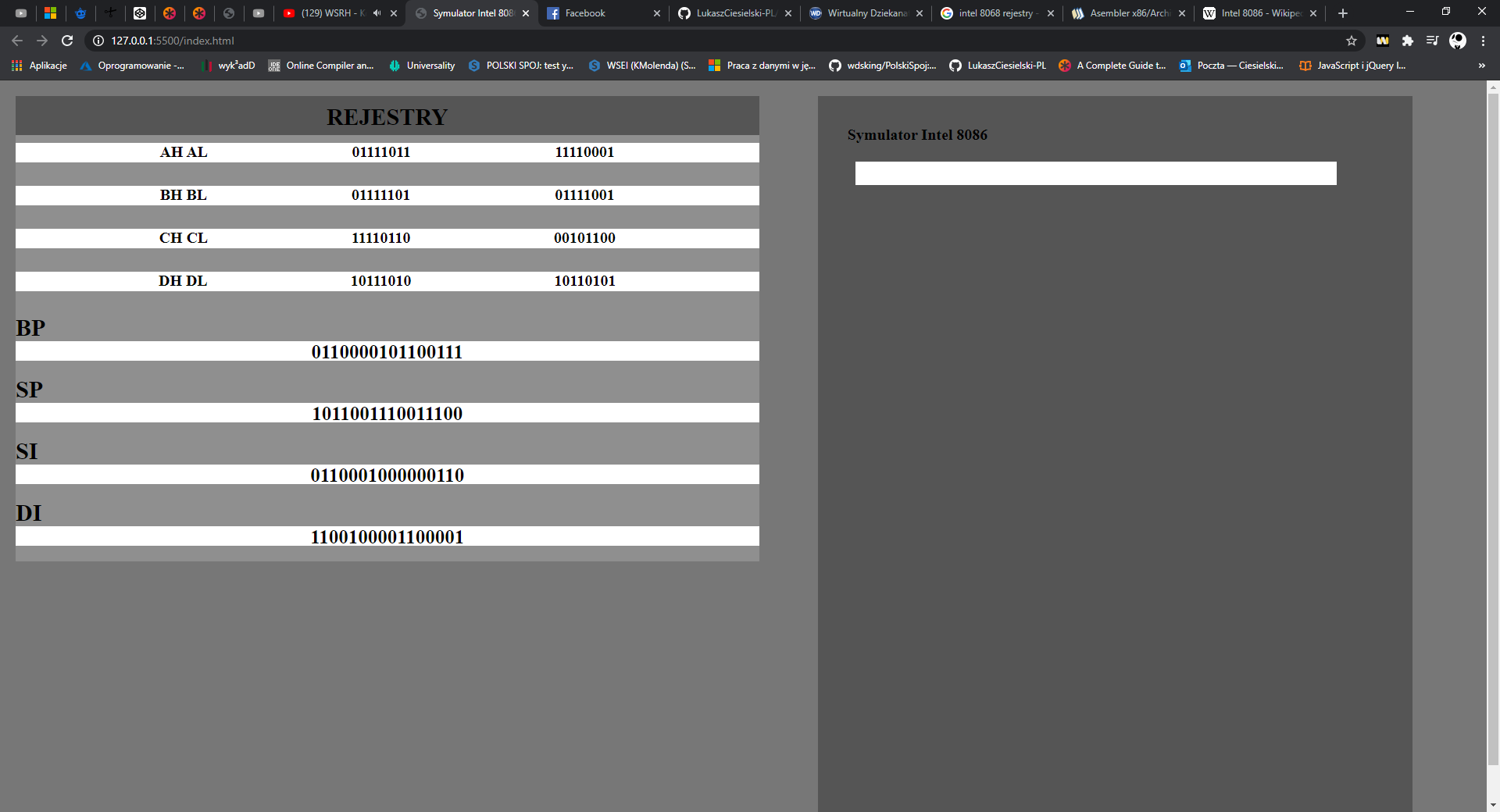
Symulacja zawiera następujące rejestry:

* **AX** –> rozdzielony na dwie 8-bitowe części AH i AL.
  + *(EAX) używany głównie w czasie operacji arytmetycznych*
* **BX** –> rozdzielony na dwie 8-bitowe części BH i BL.
  + *(EBX) rejestr bazowy, wraz z rejestrem segmentowym DS tworzą wskaźnik na przestrzeń danych*
* **CX** –> rozdzielony na dwie 8-bitowe części CH i CL.
  + *(ECX) rejestr służący głownie jako licznik w zapętlonych operacjach*
* **DX** –> rozdzielony na dwie 8-bitowe części DH i DL.
  + *(EDX) ejestr danych, używany do operacji arytmetycznych oraz obsługi wejścia-wyjścia*

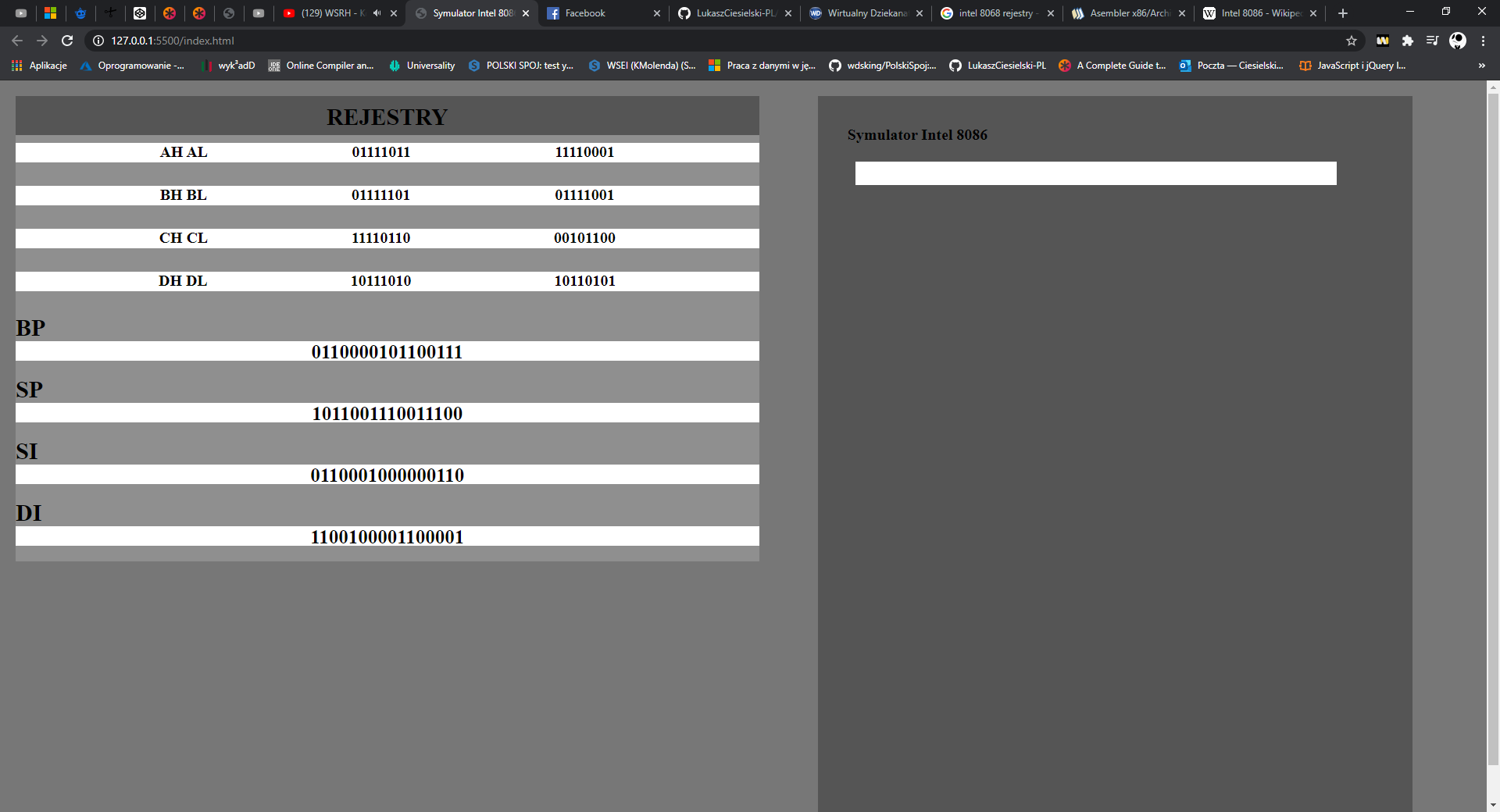


Następujące wskaźniki:

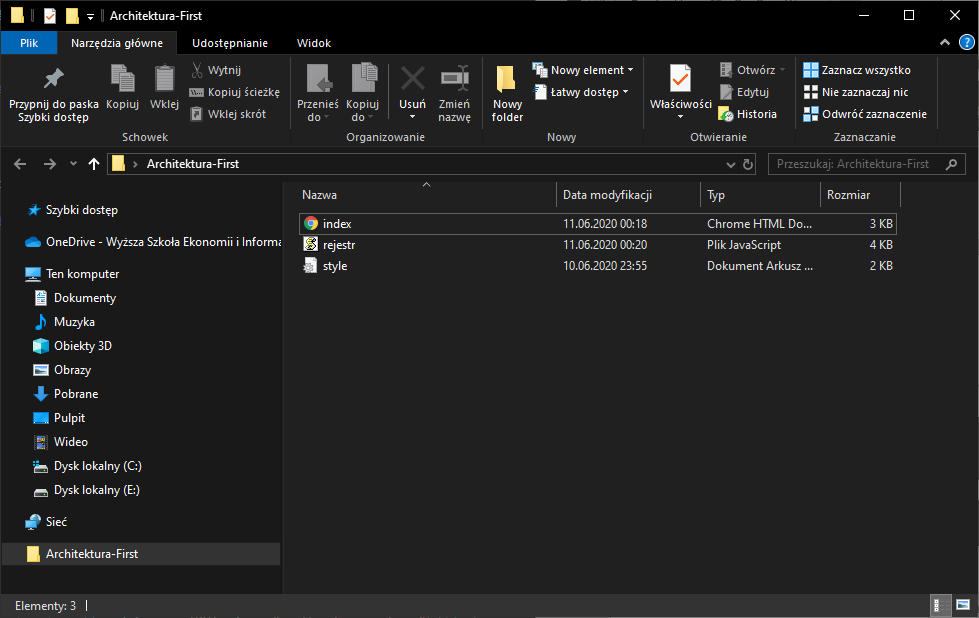
* **SP** –> 16-bitowy
  + *(ESP) wskaźnik wierzchołka stosu - 32 BITOWY*
* **BP** –> 16-bitowy
  + *(EBP) wskaźnik do danych w segmencie stosu - 32 BITOWY*
* **SI** –> 16-bitowy
  + *(ESI) rejestr indeksowy, wskaźnik źródła - 32 BITOWY*
* **DI** –> 16-bitowy
  + *(EDI) rejestr indeksowy, wskaźnik przeznaczenia - 32 BITOWY*



Oraz konsolę odpowiadającą za przesyłanie polecenia **MOV.**



Symulator został napisany z użyciem kodu **HTML**, **CSS** oraz **JS**.



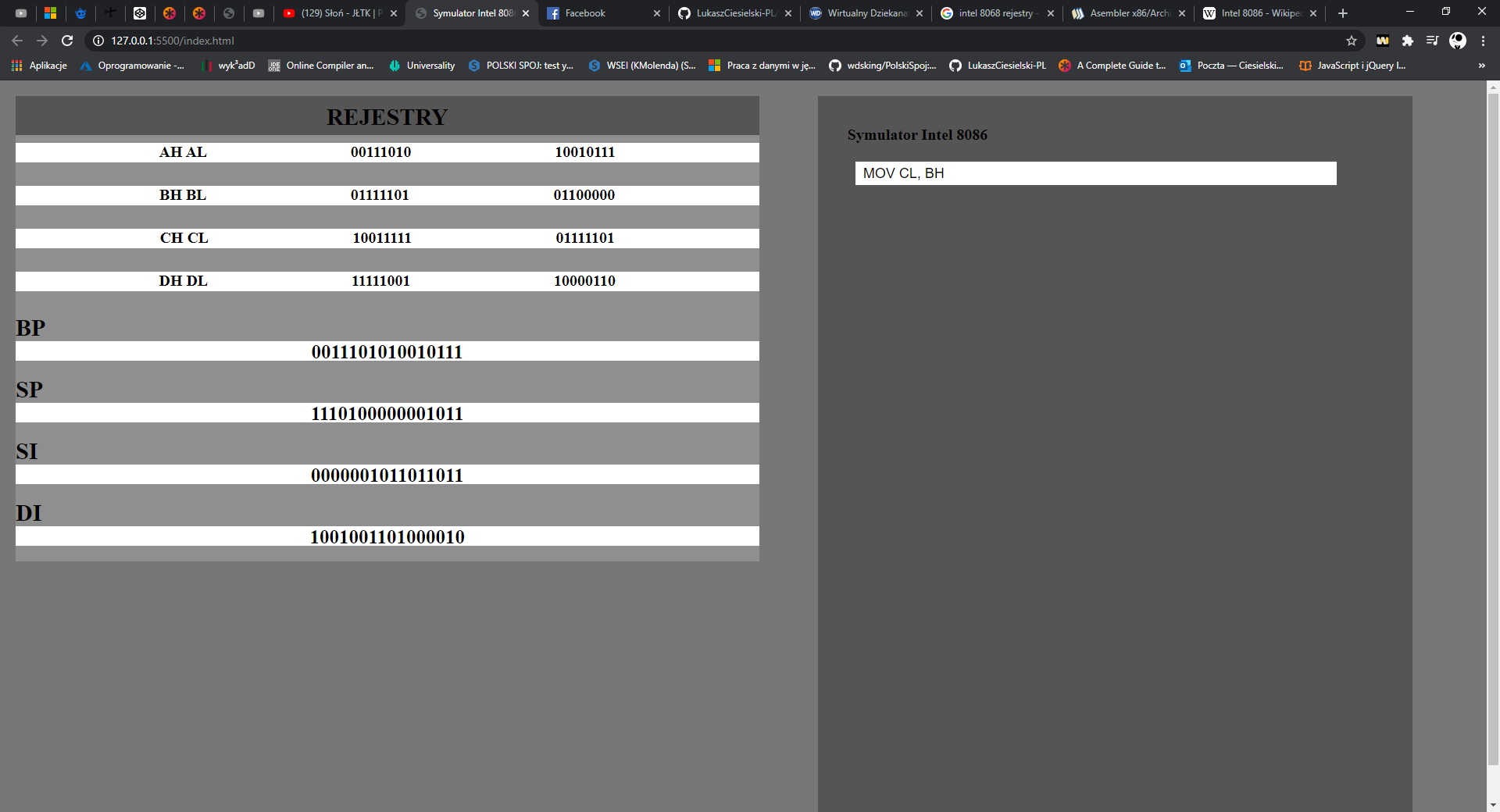
**Zasada działania**

Rejestry AX, BS, CX oraz DX zostały podzielone na dwie części 8-bitowe.

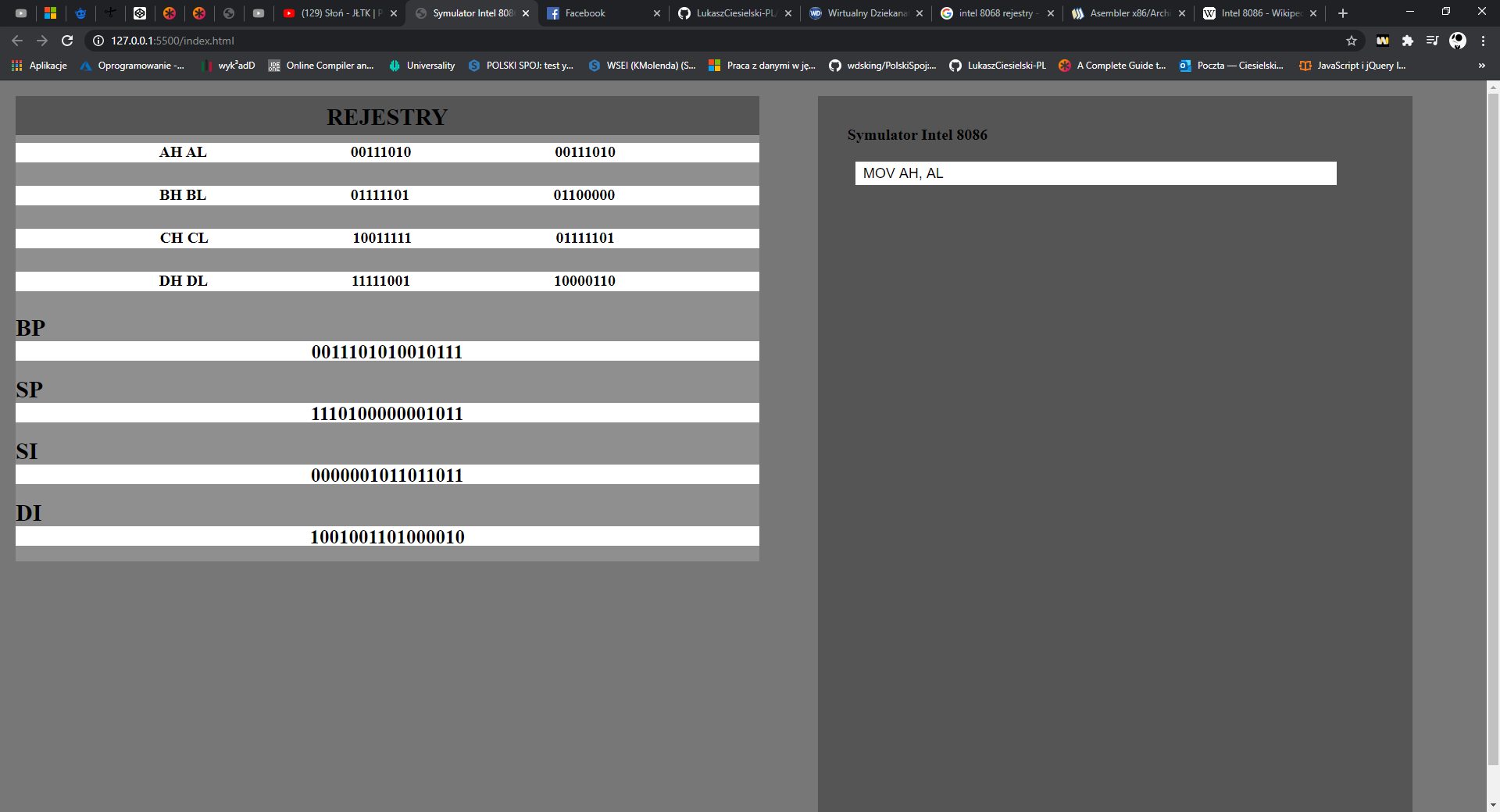
Pozwala to na przenoszenie zawartości pomiędzy nimi za pomocą polecenia MOV wewnętrznie jaki i zewnętrznie do wskaźników BP, SP, SI, DI (16-bitowe).

Wybrane przykłady pracy programu:

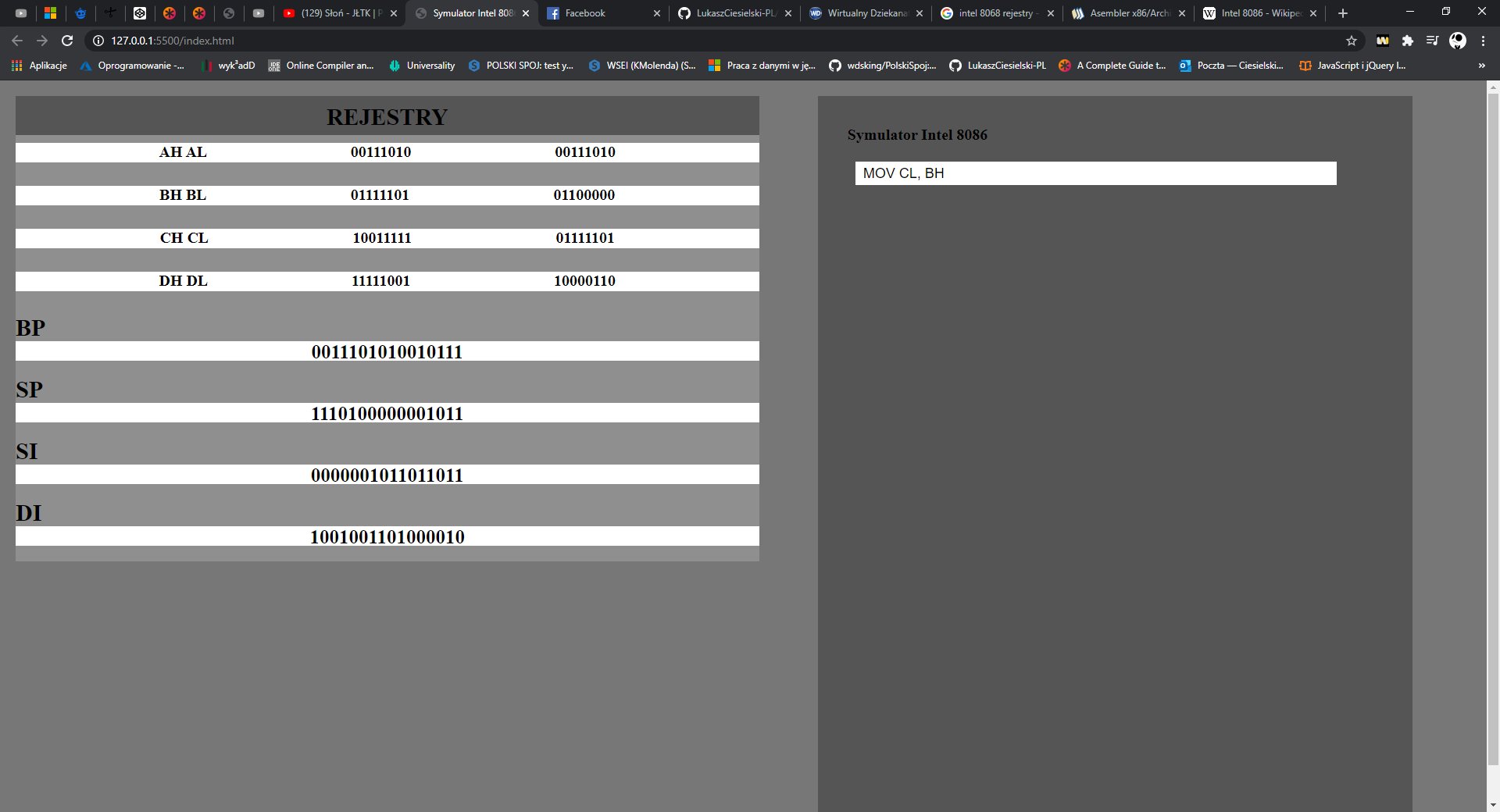
**Pierwotna forma**



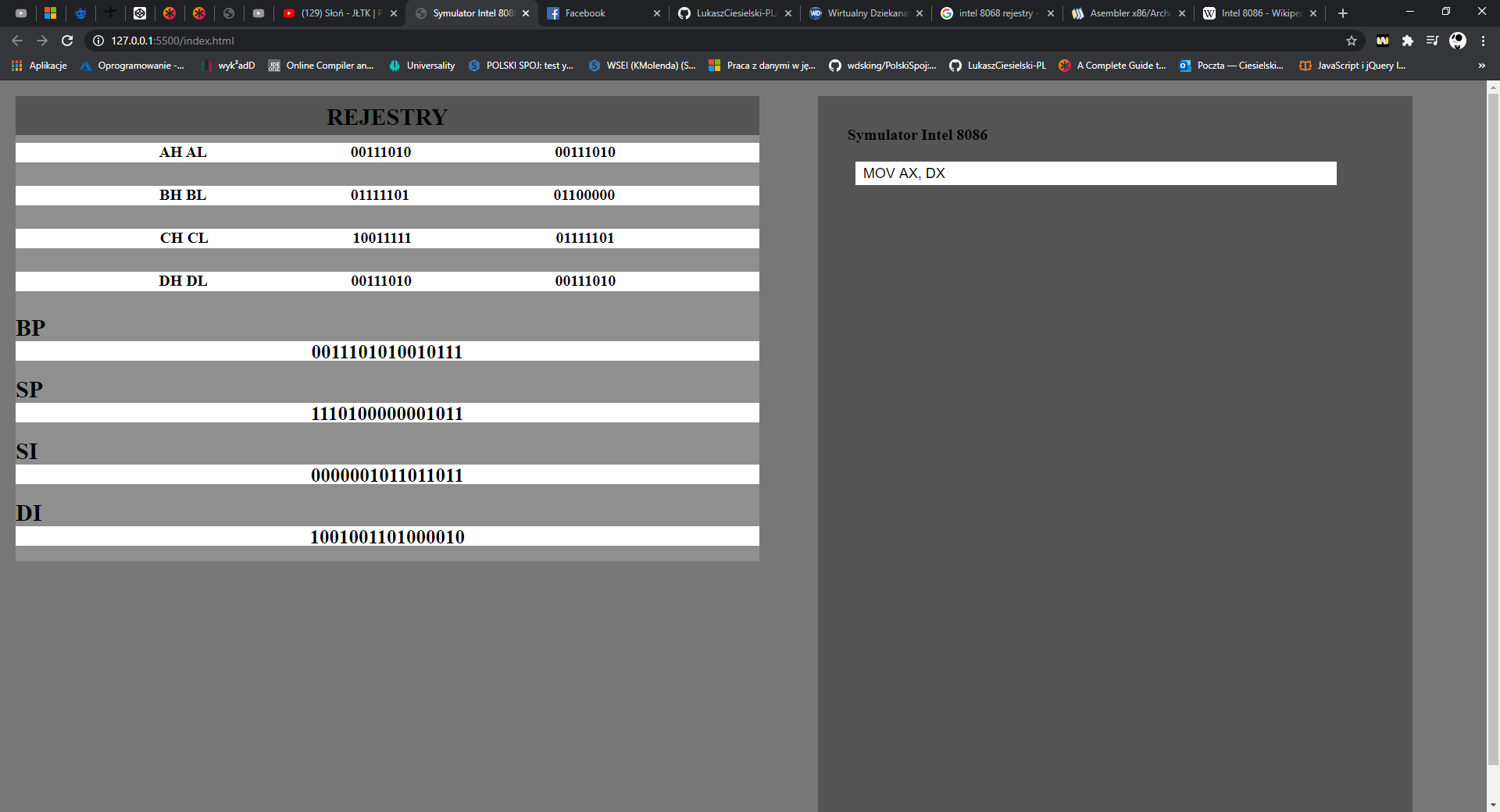
**Polecenie MOV AH, AL (przeniesienie wartości z AH do AL)**



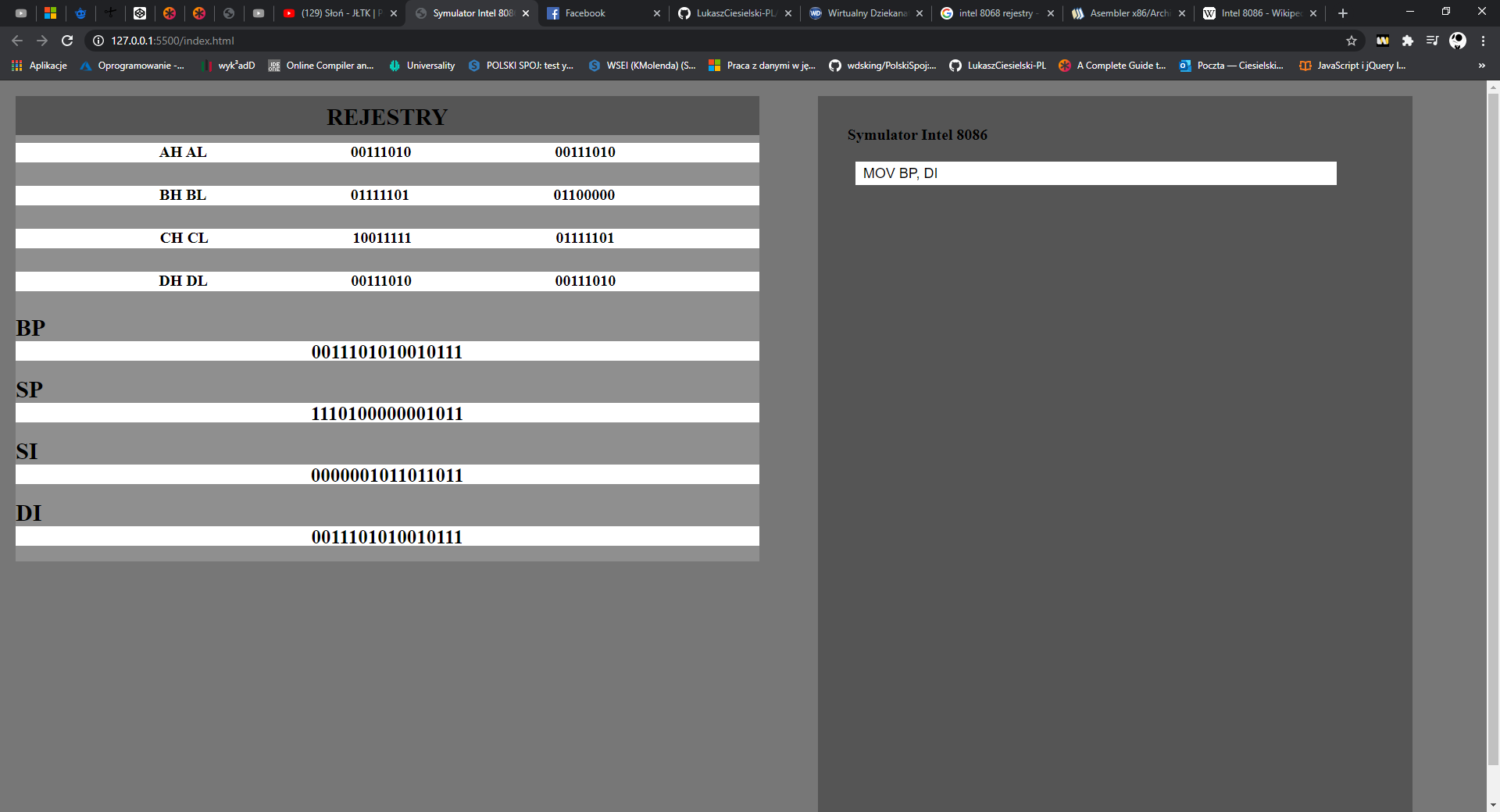
**Polecenie MOV CL, BH (przeniesienie wartości z BH do CL)**



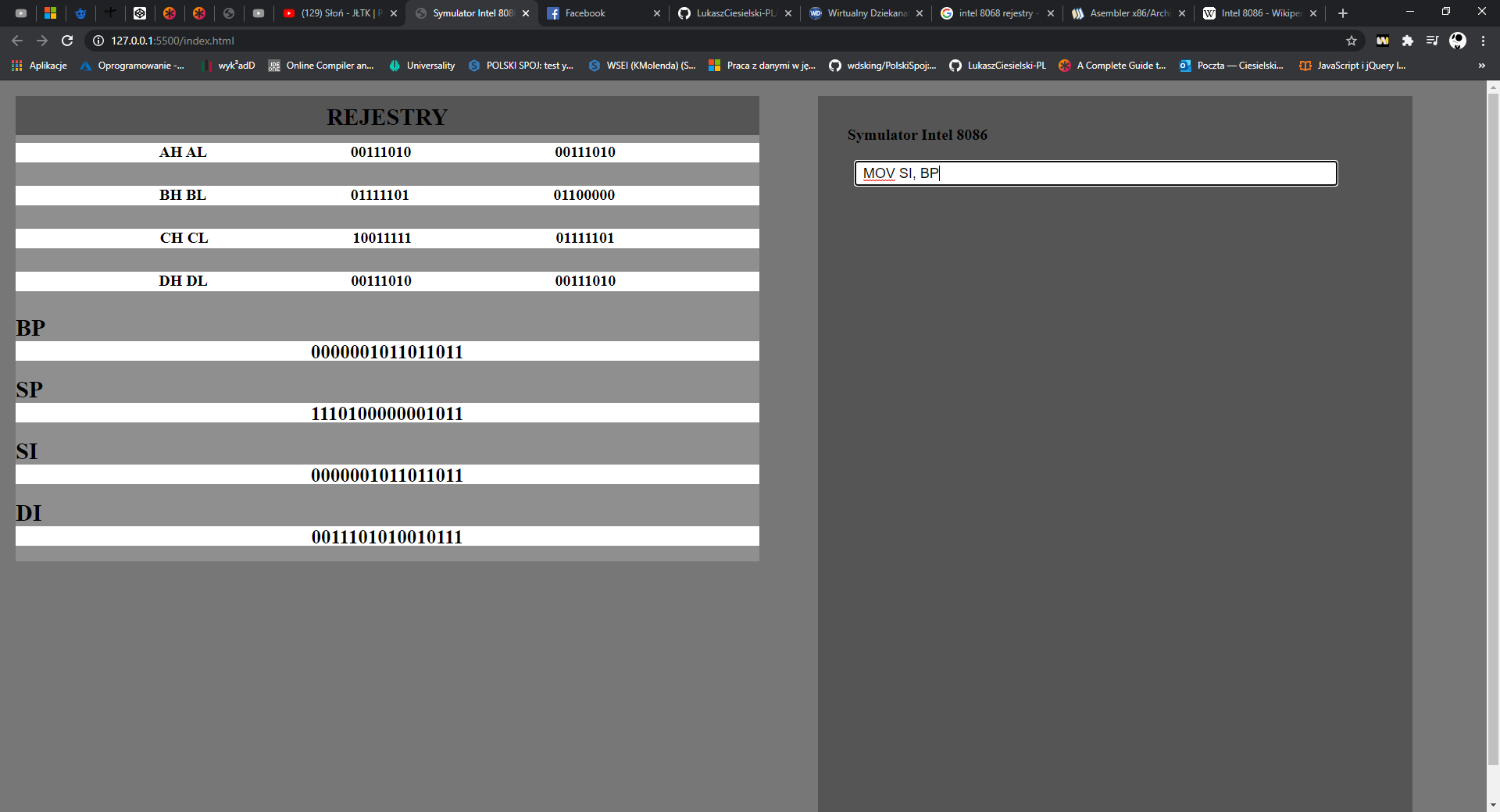
**Polecenie MOV AX, DX (przeniesienie całej wartości)**



**Polecenie MOV BP, DI (przeniesienie całej wartości)**



**Polecenie MOV SI, BP (przeniesienie całej wartości)**



**KOD ŹRÓDŁOWY**

**HTML**

<!Doctype html>

<html lang="pl">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Symulator Intel 8086</title>

    <link rel="stylesheet" href="style.css">

</head>

<body>

    <div class="kontent">

        <div class="okno" id="oknoRejestru">

            <div id="naglowek">REJESTRY</div>

            <div id="rejestr">

                <div class="rejestr"> AH AL

                    <div data-reg='AH' class="rejWysoki osemkowy"></div>

                    <div data-reg='AL' class="rejNiski osemkowy"></div>

                </div>

               <div class="rejestr"> BH BL

                    <div data-reg='BH' class="rejWysoki osemkowy"></div>

                    <div data-reg='BL' class="rejNiski osemkowy"></div>

                </div>

                <div class="rejestr"> CH CL

                    <div data-reg='CH' class="rejWysoki osemkowy"></div>

                    <div data-reg='CL' class="rejNiski osemkowy"></div>

                </div>

                <div class="rejestr"> DH DL

                    <div data-reg='DH' class="rejWysoki osemkowy"></div>

                    <div data-reg='DL' class="rejNiski osemkowy"></div>

                </div>

                <div class="kontentDolny">

                    BP

                    <div data-reg='BP' class="rejestrWy szesnastkowy"></div>

                    SP

                    <div data-reg='SP' class="rejestrWy szesnastkowy"></div>

                    SI

                    <div data-reg='SI' class="rejestrWy szesnastkowy"></div>

                    DI

                    <div data-reg='DI' class="rejestrWy szesnastkowy"></div>

                </div>

            </div>

        </div>

        <div for="adres">

            <div class="konsola">

                <div class="Wiersz">Symulator Intel 8086</div><br>

                <div class="Wiersz aktualnyWiersz"><input type="text" id="adres"></input></div>

            </div>

        </div>

    </div>

    <script src="rejestr.js" defer></script>

</html>

**CSS**

\*{

    box-sizing: border-box;

    margin: 0px;

    padding: 0px;

}

body{

    height: 105vh;

    display: flex;

}

.kontent{

    background-color: #777;

    width: 100%;

    min-height: 100vh;

}

.okno{

    background-color: #8f8f8f;

    display: flex;

    flex-direction: column;

    margin: 20px;

}

#oknoRejestru{

    width: 50%;

    height: auto;

}

#naglowek{

    width: 100%;

    height: 50px;

    background-color: #555;

    font-size: 30px;

    font-weight: 700;

    color: #000;

    display: flex;

    justify-content: center;

    margin-bottom: 10px;

    padding-top: 10px;

}

#rejestr{

    height: 100%;

    display: flex;

    flex-direction: column;

}

.rejestr{

    justify-content: space-evenly;

    background-color: #fff;

    width: 100%;

    height: 25px;

    font-size: 20px;

    font-weight: 700;

    display: flex;

    margin-bottom: 30px;

}

.kontentDolny{

    font-size: 30px;

    font-weight: 700;

}

.rejestrWy {

    background-color: #fff;

    width: 100%;

    height: 25px;

    font-size: 25px;

    text-align: center;

    color: #000;

    flex-direction: row;

    margin-bottom: 20px;

}

.konsola{

    position: absolute;

    top: 20px;

    right: 5%;

    padding: 2%;

    width: 40%;

    min-height: 100%;

    background-color: #555;

    color: #000;

    font-size: 20px;

    font-weight: 700;

}

.Wiersz>input {

    width: 90%;

    height: 30px;

    background-color: #fff;

    margin-bottom: 10px;

    margin-left: 10px;

    padding-left: 10px;

    border: 0px;

    font-size: 18px;

    color: #000;

}

**JS**

const getRegisterValue = (multiplier) => [...Array(multiplier)].map(() => Math.floor(Math.random() \* 2));

const rejestrID = {};

//rozbicie rejestrów \*x -> na 2 osobne 8 bitowe \*l i \*h

["AL", "BL", "CL", "DL", "AH", "BH", "CH", "DH"].map(code => rejestrID[code] = getRegisterValue(8));

["BP", "SP", "SI", "DI"].map(code => rejestrID[code] = getRegisterValue(16));

rejestrID.AX = rejestrID["AH"].concat(rejestrID["AL"]);

rejestrID.BX = rejestrID["BH"].concat(rejestrID["BL"]);

rejestrID.CX = rejestrID["CH"].concat(rejestrID["CL"]);

rejestrID.DX = rejestrID["DH"].concat(rejestrID["DL"]);

const rejestr = [...document.querySelectorAll('.osemkowy')];

const szesnastkowyrejestr = document.querySelectorAll('.szesnastkowy');

rejestr.forEach(el => el.innerHTML = rejestrID[el.dataset.reg].join(""));

szesnastkowyrejestr.forEach(el => el.innerHTML = rejestrID[el.dataset.reg].join(""));

const nowyWiersz = () => {

    const ostatniWiersz = document.querySelector('.aktualnyWiersz');

    const wiersz = document.querySelector('.konsola');

    const ostatniWierszwpisz = ostatniWiersz.querySelector('wpisz');

    ostatniWiersz.classList.remove('aktualnyWiersz');

    ostatniWierszwpisz.disabled = true;

    ostatniWierszwpisz.id = "";

    let linia = document.createElement("div");

    linia.className = "Wiersz aktualnyWiersz";

    linia.innerHTML = `<wpisz type="text" id="adres"></wpisz>`;

    wiersz.appendChild(linia);

}

document.addEventListener('keypress', e => {

    if (e.keyCode == 13) {

        const dane = document.querySelector('#adres');

        const wpisz = dane.value.toUpperCase();

        const commands = wpisz.replace(',', ' ').split(' ').filter(el => el != "");

        if (commands[0].toUpperCase() == "MOV" && rejestrID[commands[2]].length == rejestrID[commands[1]].length) {

            rejestrID[commands[2]] = rejestrID[commands[1]];

            if (commands[2].includes("X") && commands[1].includes("X")) {

                rejestrID[commands[2].replace("X", "H")] = rejestrID[commands[2]].slice(0, 8);

                rejestrID[commands[2].replace("X", "L")] = rejestrID[commands[2]].slice(8, 16);

                rejestr.forEach(el => el.innerHTML = rejestrID[el.dataset.reg].join(""));

                szesnastkowyrejestr.forEach(el => el.innerHTML = rejestrID[el.dataset.reg].join(""));

            } else if (commands[2].includes("X")) {

                rejestrID[commands[2].replace("X", "H")] = rejestrID[commands[2]].slice(0, 8);

                rejestrID[commands[2].replace("X", "L")] = rejestrID[commands[2]].slice(8, 16);

                rejestr.forEach(el => el.innerHTML = rejestrID[el.dataset.reg].join(""));

                szesnastkowyrejestr.forEach(el => el.innerHTML = rejestrID[el.dataset.reg].join(""));

            } else {

                if (rejestrID[commands[2]].length == 8) {

                    let regId = commands[2].charAt(0);

                    let regszesnastkowyId = commands[2].replace("H", "X").replace("L", "X");

                    rejestrID[regszesnastkowyId] = rejestrID[`${regId}H`].concat(rejestrID[`${regId}L`]);

                }

                rejestr.forEach(el => el.innerHTML = rejestrID[el.dataset.reg].join(""));

                szesnastkowyrejestr.forEach(el => el.innerHTML = rejestrID[el.dataset.reg].join(""));

            }

            nowyWiersz();

        }

    }

});