

Mikroprocesory i mikrokontrolery Laboratorium nr 2	Temat: Obsługa portów wejścia/wyjścia mikrokontrolera 8051
Grupa: 21b	Michał Lechowicz Mateusz Moneta

Cel ćwiczenia:

Celem ćwiczenia było zapoznanie się z obsługą portów wejścia / wyjścia w procesorze Intel 8051 i w zestawie ZM2MCS51.

Zadanie na ocenę 5.0:

1. Opis zadania:

W zadaniu na 5.0 należało napisać program, który testuje refleks użytkownika. Diody od D0 do D3 włączają się w losowej kolejności na krótki odstęp czasu. Program startuje od małego okresu wyświetlania diody. Każde N prób powoduje zwiększenie czasu wyświetlania. Jeśli w trakcie świecenia diody użytkownik wskaże przycisk o numerze jej odpowiadającym, to wyświetlany jest sygnał zakończenia (zwycięstwa), jako wybrana sekwencja diod. Po wszystkim program kończy działanie.

2. Kod programu:

```
; stale dla refleksu
startWaitingTime EQU 0x60 ; startowy czas swiecenia diody

; stale dla generatora
j EQU 7 ; definujemy stala j
kSize EQU 10 ; definujemy stala k
m EQU 4 ; <0,3> ; definujemy stala m odpowiedzialna za zakres generowanych liczb
indexModuloStart EQU 3 ; definujemy pierwsza wartosc indeksu

ORG 0020H ; ustaw miejsce w kodzie na 20h
k0 EQU 4 ; definujemy wartosci poczatkowe tablicy
k1 EQU 5
k2 EQU 3
k3 EQU 3
k4 EQU 3
k5 EQU 4
k6 EQU 6
k7 EQU 2
k8 EQU 6
k9 EQU 5

; komorki pamieci dla refleksu
waitTime DATA 60h ; aktualny czas swiecenia

; komorki pamieci dla generatora
poczatekTablicy DATA 20h ; definujemy poczatek tablicy
koniecTablicy DATA 29h ; definujemy koniec tablicy
first_element DATA 32h ; definujemy pierwszy element dla wartosci modulo
second_element DATA 33h ; definujemy drugi element dla wartosci modulo
result DATA 34h
Modulo1 DATA 40h ; pierwsza czesc wyrazenia modulo
Modulo2 DATA 41h ; druga czesc wyrazenia modulo
indexModulo DATA 42h
Quotient DATA 50h ; czesc calkowita z dzielenia
Remainder DATA 51h ; reszta z dzielenia

CSEG AT 0
    ACALL INIT_GEN
    LJMP INIT_REFLEX

CSEG AT 100h
;-----GENERATOR-----

    RUN_GENERATOR:
        ACALL INDEKS_1 ; oblicz indeks i-j+k
        ACALL GET_FROM_ARRAY ; pobierz wartosci z rejestru cyklicznego
        ACALL SUM ; dodaj dwie wartosci pobrane z rejestru cyklicznego
        ACALL CALCULATE_RANDOM ; oblicza wartosc losowa i przenosi ja do akumulatora

        RET ; powrot z funkcji

    INIT_GEN: ; wpisanie wartosci tablicy do pamieci RAM
        MOV poczatekTablicy, #k0
        MOV 21h, #k1
        MOV 22h, #k2
        MOV 23h, #k3
```

```
MOV 24h, #k4
MOV 25h, #k5
MOV 26h, #k6
MOV 27h, #k7
MOV 28h, #k8
MOV koniecTablicy, #k9
```

```
ACALL RESET_INDEXES
RET
```

MODULO:

```
MOV A, Modulo1 ; prznies pierwszy argument z Modulo1 do A
MOV B, Modulo2 ; prznies pierwszy argument z Modulo1 do B
```

```
DIV AB ; Dziel A przez B
```

```
MOV Quotient, A; Zapisz czesc calkowita do komorki RAM 50H
MOV Remainder, B; Zapisz reszte do komorki RAM 51H
```

```
RET ; powrot z funkcji
```

INDEKS_1:

```
MOV Modulo1, indexModulo ; 3 = pierwotna wartosc indeksu dla k=10, prznies z indexModulo do Modulo1
MOV Modulo2, #kSize ; mod k, gdzie k = 10; prznies wartosc kSize do Modulo2
```

```
ACALL MODULO ; wywołanie funkcji modulo
```

```
MOV A, Remainder ; zapisz reszte z dzielenia (wynik funkcji modulo) do A
ADD A, #20h ; dodaj 20h do akumulatora by uzyskac indeks
MOV R0, A ; zapisz wartosc z akumulatora do rejestru R0
```

```
RET ; powrot z funkcji
```

GET_FROM_ARRAY:

```
MOV first_element, @R0 ; pobierz pierwsza skladowa do pamieci RAM (adresowanie posrednie)
MOV second_element, @R1 ; pobierz druga skladowa do pamieci RAM (adresowanie posrednie)
```

```
RET ; powrot z funkcji
```

SUM:

```
MOV A, first_element ; prznies pierwszy element do akumulatora
ADD A, second_element ; dodaj drugi element do akumulatora
MOV result, A ; prznies wartosc z akumulatora do pamieci RAM
```

```
RET ; powrot z funkcji
```

INCREMENT:

```
INC R1 ; podnies o 1 wartosc R1 (Indeks)
INC indexModulo ; podnies o 1 wartosc indexModulo
```

```
RET ; powrot z funkcji
```

CALCULATE_RANDOM:

```
MOV Modulo1, result ; przenies result do Modulo1
MOV Modulo2, #m ; prznies wartosc parametru m do Modulo2
```

```
ACALL MODULO ; wywołuje funkcje modulo
MOV A, B ; przenies wynik funkcji modulo (szukana liczba losowa) do akumulatora
```

```
MOV @R1, A ; zapisz wartosc A do rejestru cyklicznego pod wartosc indeksu wskazywanego przez R1 (adresowanie posrednie)
```

```

        CJNE R1, #koniecTablicy, INCREMENT ; sprawdz czy indeks osiagnal koniecTablicy,
                                           ; jesli tak idz dalej, jesli nie wywolaj funkcje
INCREMENT
        SJMP RESET_INDEXES ; idz do funkcji RESET_INDEKS

        RET ; powrot z funkcji

RESET_INDEXES:
        MOV indexModulo, #indexModuloStart ; przenies wartosc 3 do indexModulo
        MOV R1, #20h ; ustaw 20 jako wartosc rejestru R1 (początek tablicy)

        RET ; powrot z funkcji

; -----REFLEKS-----

INIT_REFLEX:
        MOV waitTime, #startWaitingTime ; laduj do pamieci domyslna wartosc czasu swiecenia diody

GENERATE_RANDOM_NUMBER_OF_DIODE:
        ACALL RUN_GENERATOR ; wylosuj nowa liczbe i zapisz ja do akumulatora

RESET_PROGRAM:
        MOV P3, #0xFF ; zeruj wcisniete przyciski

SWITCH_DIODE: ; funkcja sluzaca do wlaczania diod
        ACALL CALCULATE_DIODE_TO_DISPLAY ; wywolaj funkcje obliczajaca ktora dioda ma zostac zapalona
        MOV R4, A ; wpisz linie wylosowanej diody do rejestru R4
        XRL A, #0xFF ; wykonaj alternatywe wykluczajaca, by zapalic konkretne diody
        MOV P2, A ; wpisz wartosc akumulatora do linii P2

CZEKAJ: ; funkcja czekaj na reakcje uzytkownika
        MOV R5, waitTime

        L1: NOP ; operacja NOP 1-cyklowa zegara
            NOP
            MOV R6, waitTime ; operacja 2-cyklowa
            MOV R6, waitTime

            L2: NOP
                NOP
                MOV R7, waitTime
                MOV R7, waitTime
                ; Petla L3 zuzyje 255*3 cykli maszynowych
                L3: NOP ; instrukcja 1-cyklowa
                    NOP

                DJNZ R7, L3 ; instrukcja 2-cyklowa
            DJNZ R6, L2
        DJNZ R5, L1

CHECK_ANSWER:
        MOV A, P3 ; zczytaj wartosc klawiszy z linii P3
        XRL A, #0xFF ; wykonaj XOR by sprawdzic, czy jakis klawisz jest wcisniety

        ANL A, R4 ; wykonaj iloczyn logiczny zapalonej diody oraz wcisnietego klawisza, wynik zapisz do A

        JNZ WIN ; jesli wynik rozny od 0 idz do funkcji WIN, jesli nie idz dalej

WRONG_ANSWER:

```

```

ACALL ADD_MORE_TIME

MOV ACC.0, C ; sprawdź czy wystąpiło przepełnienie (Jeśli carry flag ustawione na 1, ustaw A na 1)
JC LOSE ; jeśli tak zakończ grę, jeśli nie losuj kolejną diodę

AJMP GENERATE_RANDOM_NUMBER_OF_DIODE ; wylosuj nową diodę do zaświecenia

LOSE: ; Jeśli osiągnięta maksymalny dostępny czas reakcji i użytkownik nie dokonał poprawnego wyboru
MOV P2, #0 ; sekwencja porażki
JMP END_PROGRAM ; idź do end program

WIN:
MOV P2, #010101010b ; zapal sekwencje zwycięstwa

END_PROGRAM: ; Zakończ program
JMP END_PROGRAM

ADD_MORE_TIME: ; funkcja dodająca więcej czasu na reakcję
MOV A, waitTime ; przenieś wartość waitTime do akumulatora
ADD A, #20h ; dodaj 20h do akumulatora
MOV waitTime, A ; zapisz wartość z akumulatora do waitTime (RAM)
RET ; powrót z funkcji

CALCULATE_DIODE_TO_DISPLAY: ; konwertuj numer zaświeconej diody na kod binarny dla portu 2
DIODE3:
CJNE A, #3, DIODE2 ; jeśli wartość w A równa 3
MOV A, #1000b ; zapisz kod binarny dla diody czwartej, jeśli nie idź dalej
RET ; wyjdź z funkcji
DIODE2:
CJNE A, #2, DIODE1 ; jeśli wartość w A równa 2
MOV A, #0100b
RET
DIODE1:
CJNE A, #1, DIODE0 ; jeśli wartość w A równa 1
MOV A, #0010b
RET
DIODE0: ; jeśli wartość różna od 3 lub 2 lub 1
MOV A, #0001b ; wpisz kod binarny dla diody numer 0
RET ; powrót z funkcji

```

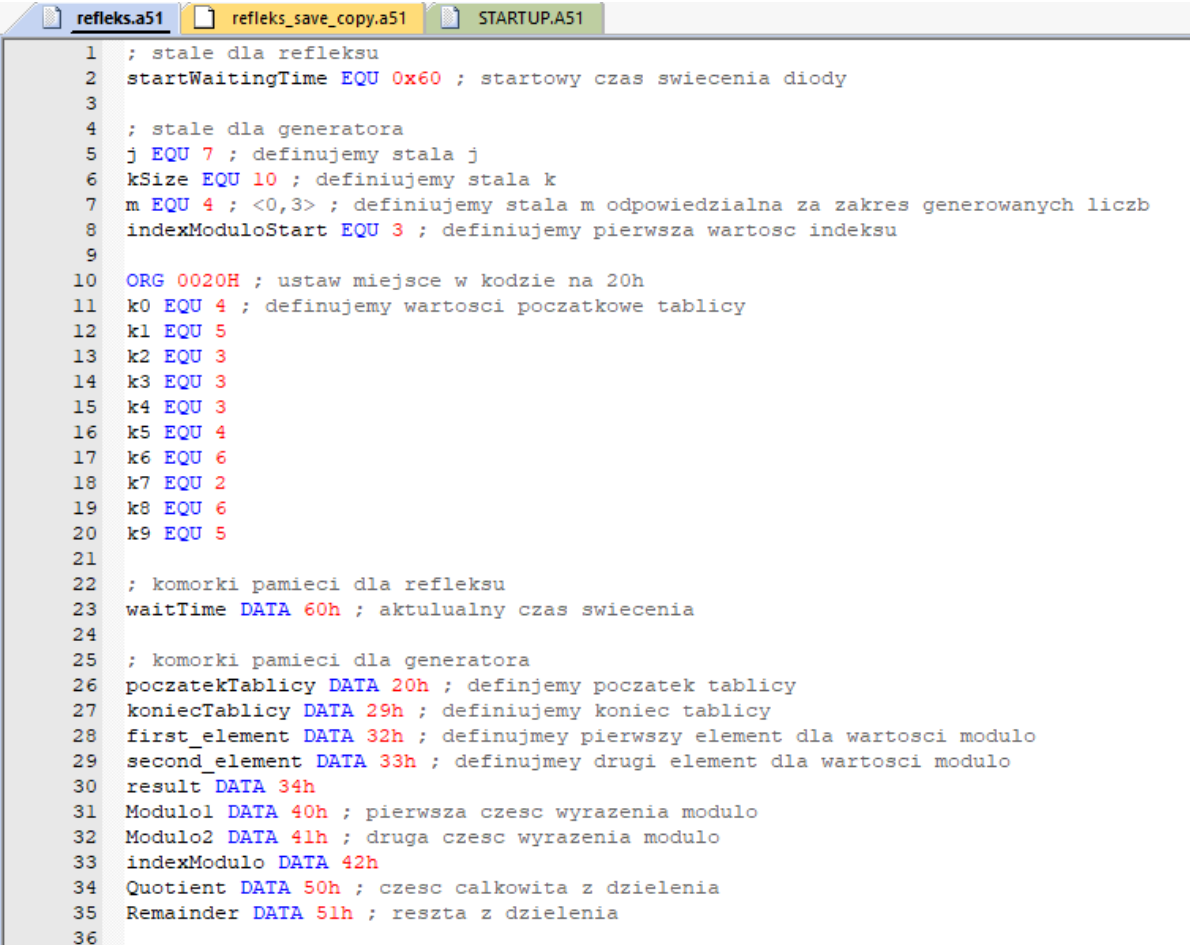
END

Kod programu

Powyższy kod przedstawia program do testowania refleksu, w którym wykorzystano generator liczb pseudolosowych, napisany w Assemblerze dla platformy Intel 8051.

3. Opis programu:

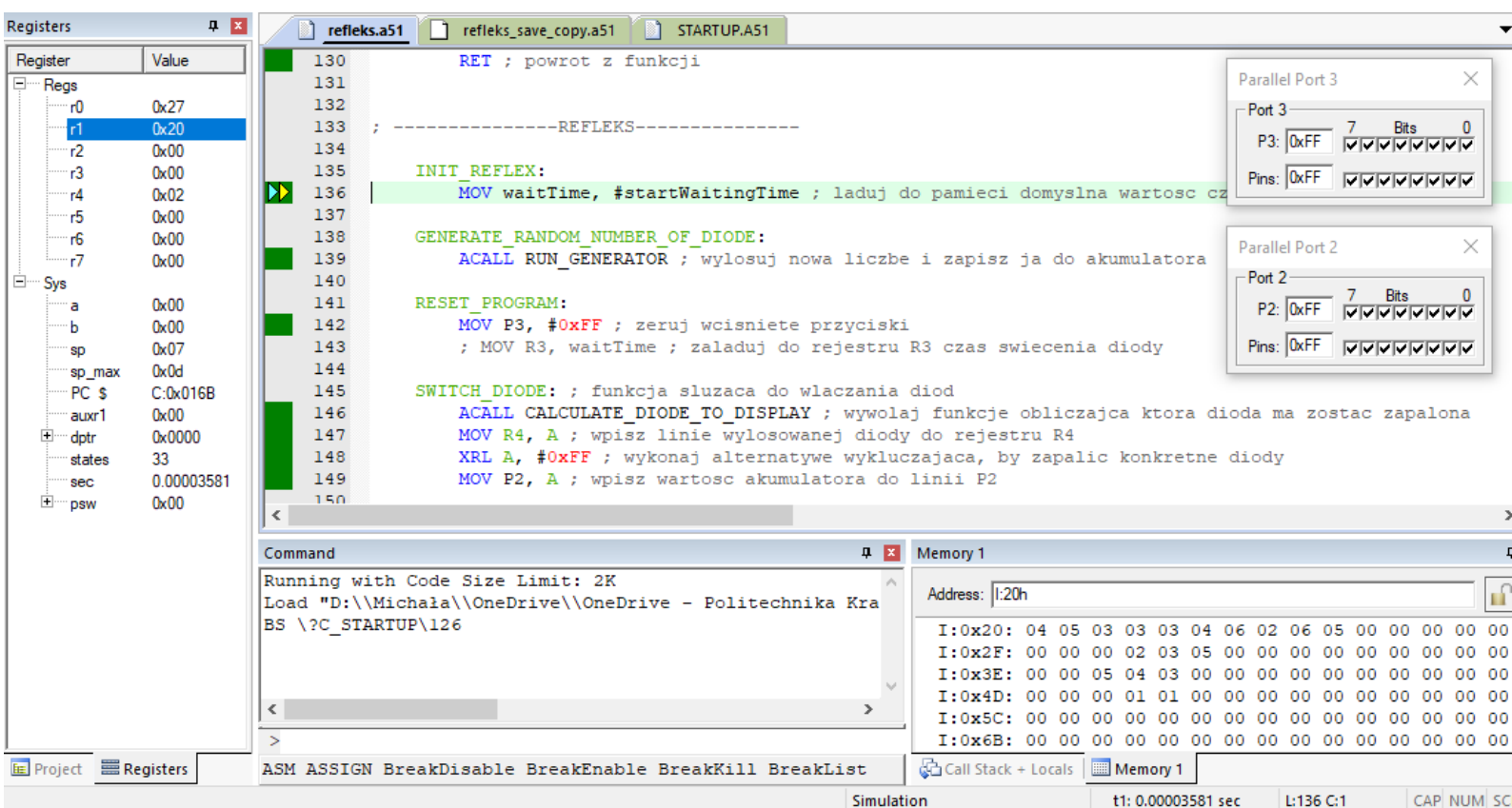
0. Dla zwiększenia czytelności w opisie pominięto wytłumaczenie części programu odpowiedzialnej za generator Fibonnaciego, gdyż zostało to wykonane w laboratorium nr 1;
1. Na początku programu definiujemy stałą w pamięci aplikacji, która przechowuje domyślny czas świecenia diody. Następnie definiujemy miejsce w pamięci RAM dla czasu świecenia diody;



```
1 ; stale dla refleksu
2 startWaitingTime EQU 0x60 ; startowy czas swiecenia diody
3
4 ; stale dla generatora
5 j EQU 7 ; definujemy stala j
6 kSize EQU 10 ; definujemy stala k
7 m EQU 4 ; <0,3> ; definujemy stala m odpowiedzialna za zakres generowanych liczb
8 indexModuloStart EQU 3 ; definujemy pierwsza wartosc indeksu
9
10 ORG 0020H ; ustaw miejsce w kodzie na 20h
11 k0 EQU 4 ; definujemy wartosci poczatkowe tablicy
12 k1 EQU 5
13 k2 EQU 3
14 k3 EQU 3
15 k4 EQU 3
16 k5 EQU 4
17 k6 EQU 6
18 k7 EQU 2
19 k8 EQU 6
20 k9 EQU 5
21
22 ; komorki pamieci dla refleksu
23 waitTime DATA 60h ; aktualny czas swiecenia
24
25 ; komorki pamieci dla generatora
26 poczatekTablicy DATA 20h ; definujemy poczatek tablicy
27 koniecTablicy DATA 29h ; definujemy koniec tablicy
28 first_element DATA 32h ; definujemy pierwszy element dla wartosci modulo
29 second_element DATA 33h ; definujemy drugi element dla wartosci modulo
30 result DATA 34h
31 Modul01 DATA 40h ; pierwsza czesc wyrazenia modulo
32 Modul02 DATA 41h ; druga czesc wyrazenia modulo
33 indexModulo DATA 42h
34 Quotient DATA 50h ; czesc calkowita z dzielenia
35 Remainder DATA 51h ; reszta z dzielenia
36
```

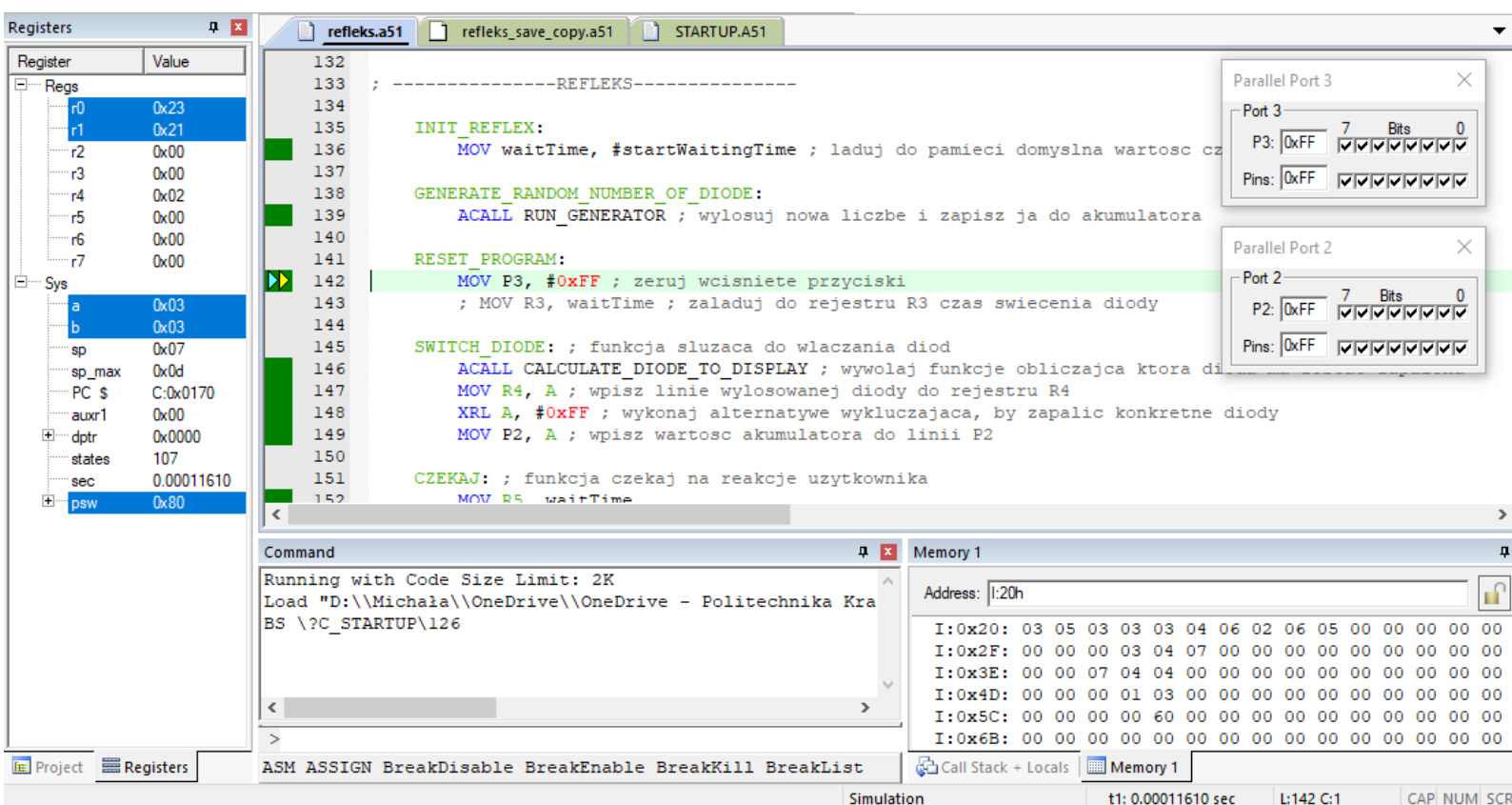
Rys. 1. Definicja stałych i zmiennych

- Następnie inicjalizujemy program za pomocą funkcji `INIT_REFLEX`. Ładujemy wartość startową czasu świecenia diody do pamięci RAM;



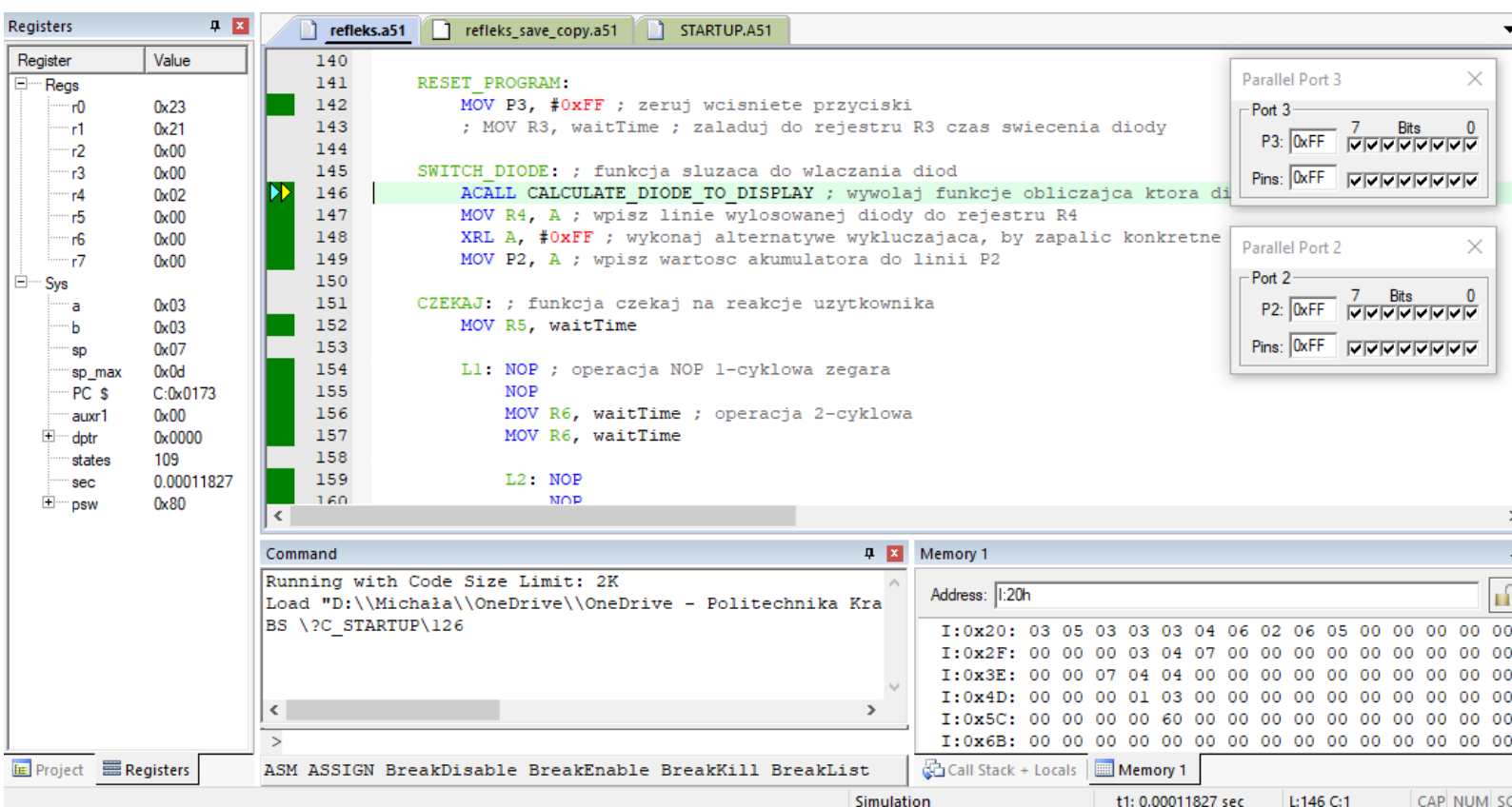
Rys. 2. Zapis waitTime do pamięci RAM

3. Dalej generowana jest nowa liczba do rejestru A, za pomocą generatora Fibonnaciego;



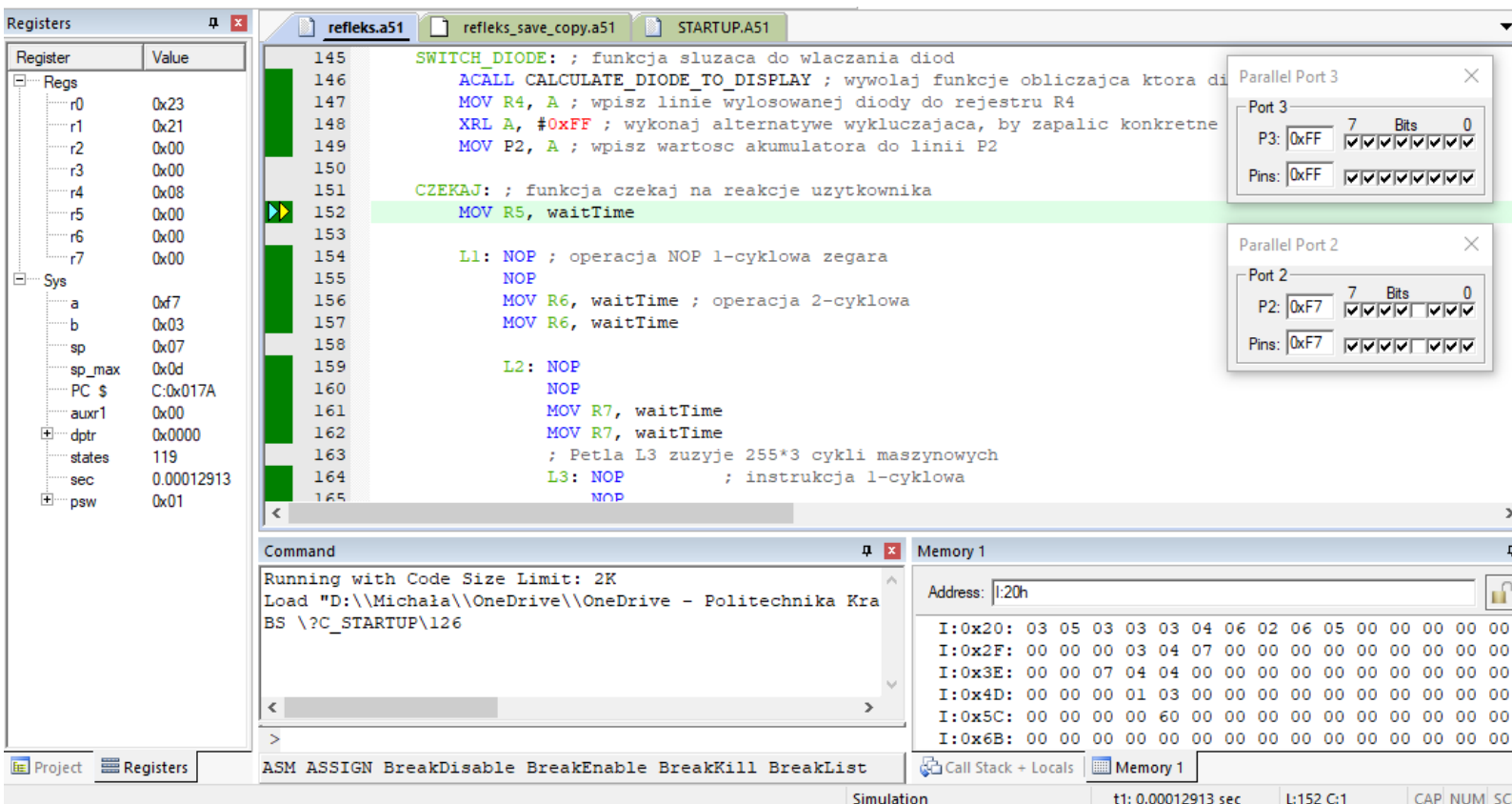
Rys. 3. Generowanie nowej liczby pseudolosowej

4. Następnie resetowane są wciśnięte klawisze dla portu **P3**;

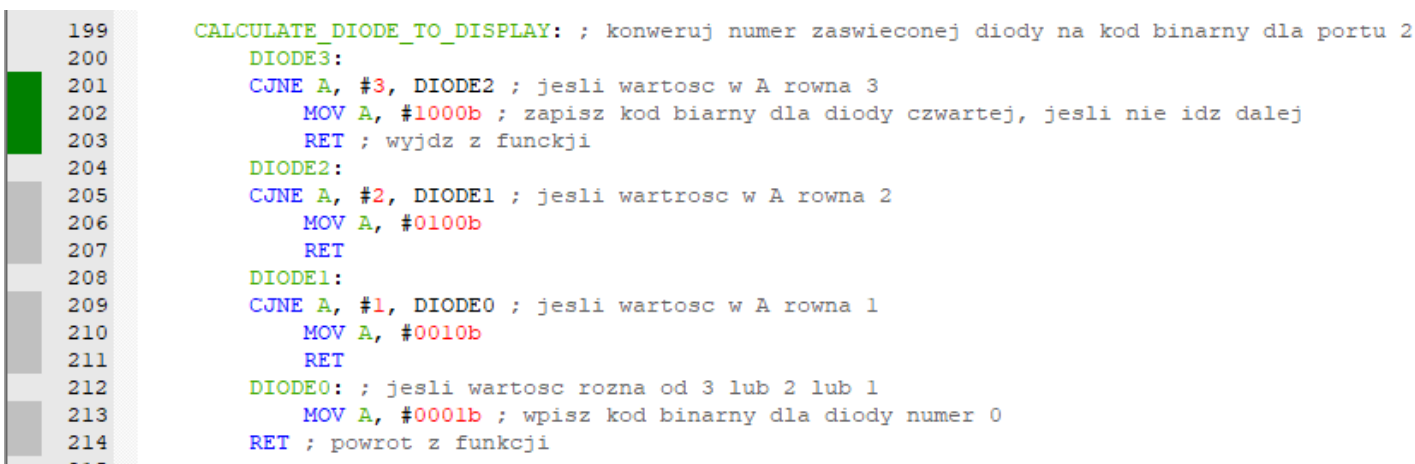


Rys. 4. Resetowanie ustawień

- Dalej włączana jest konkretna dioda, według numeru, który został wylosowany przez generator. W tym celu używana jest funkcja **CALCULATE_DIODE_TO_DISPLAY**, która przelicza wartość dziesiętną z generatora, na ciąg bitów dla portu **P2**;

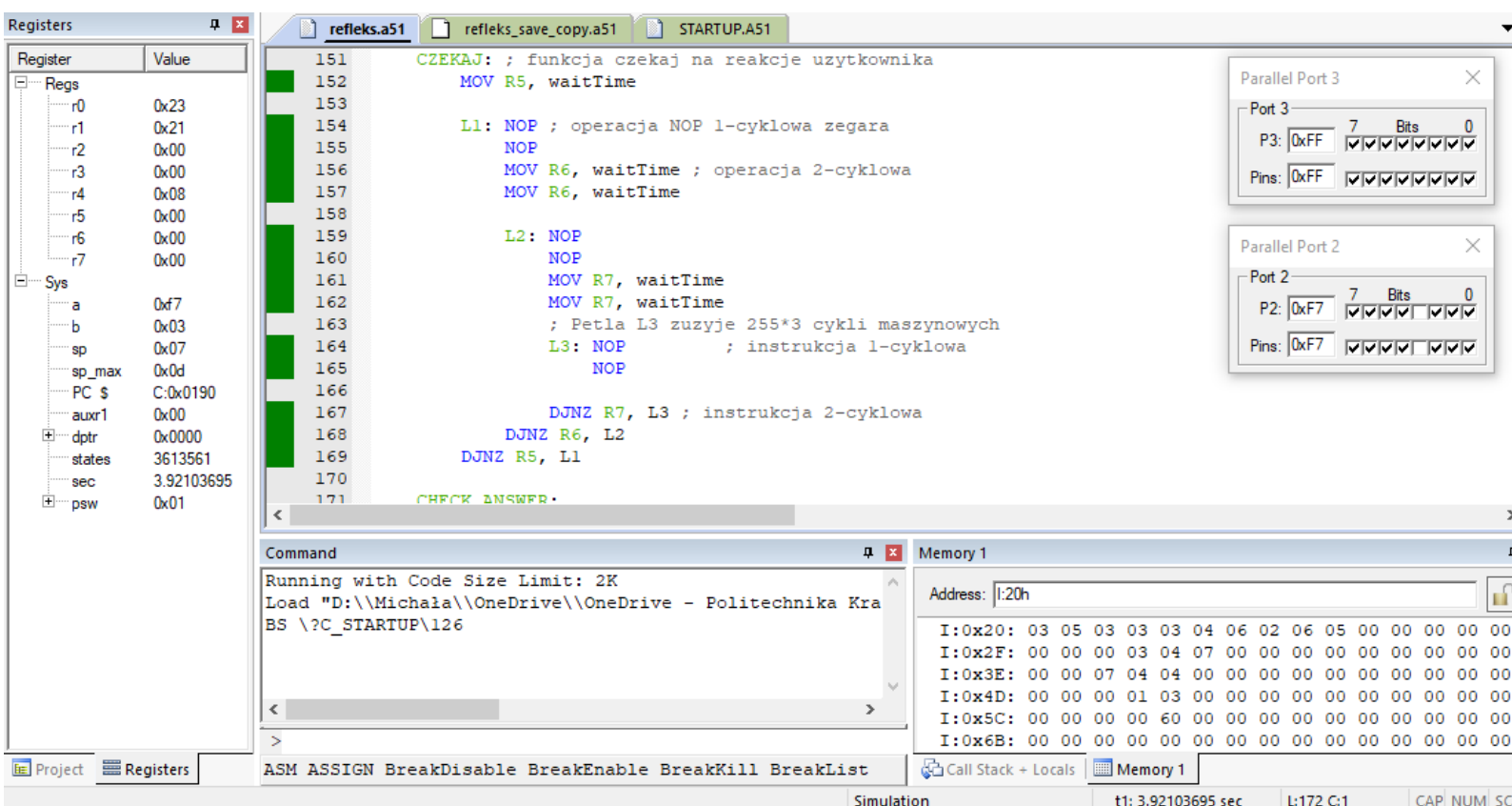


Rys. 5. Włączanie wylosowanej diody



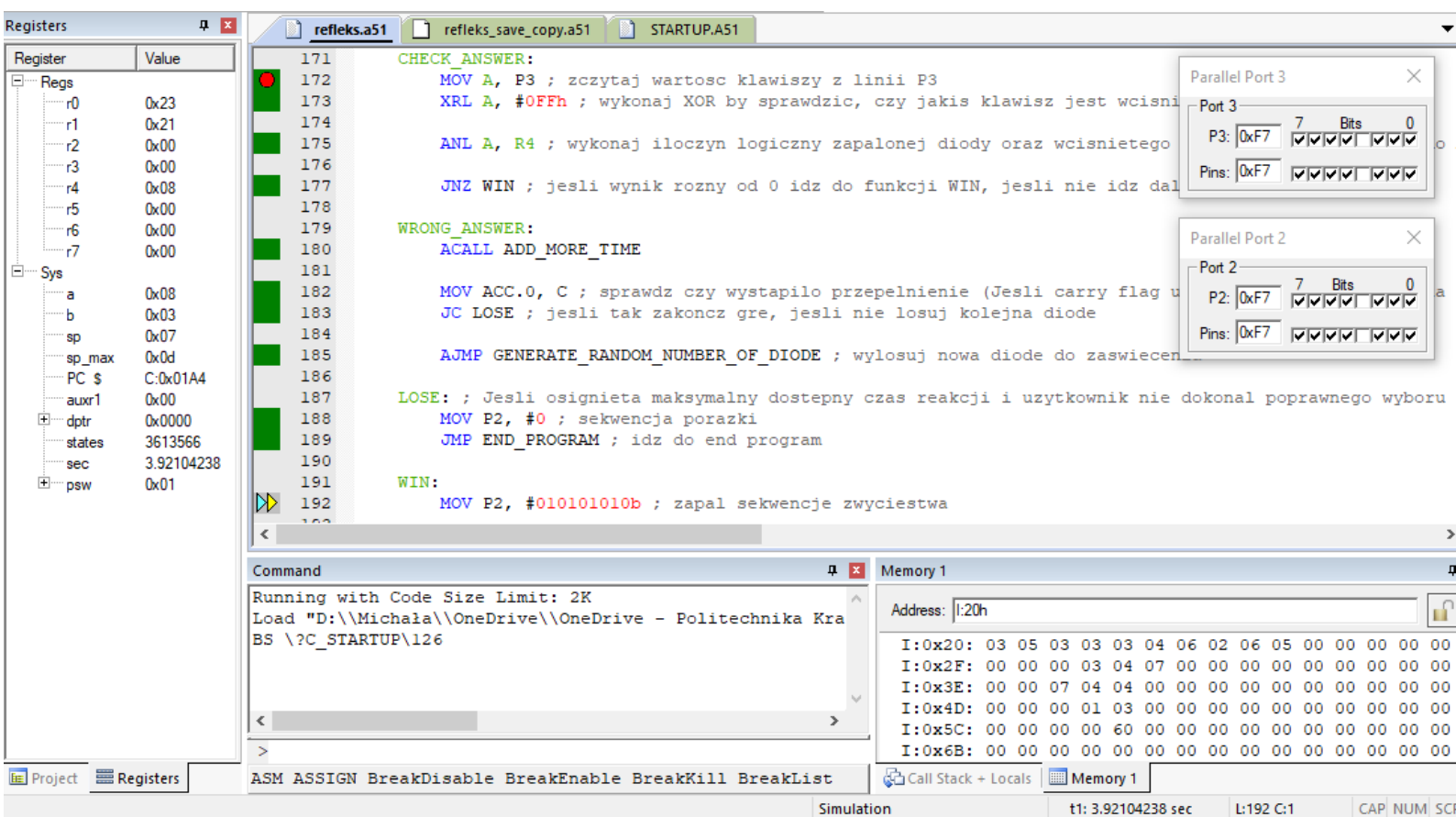
Rys. 6. Konwertowanie wylosowanej diody na kod binarny dla portu **P2**

- Po włączeniu diody przechodzimy do etykiety CZEKAJ, w której program czeka na reakcję użytkownika;



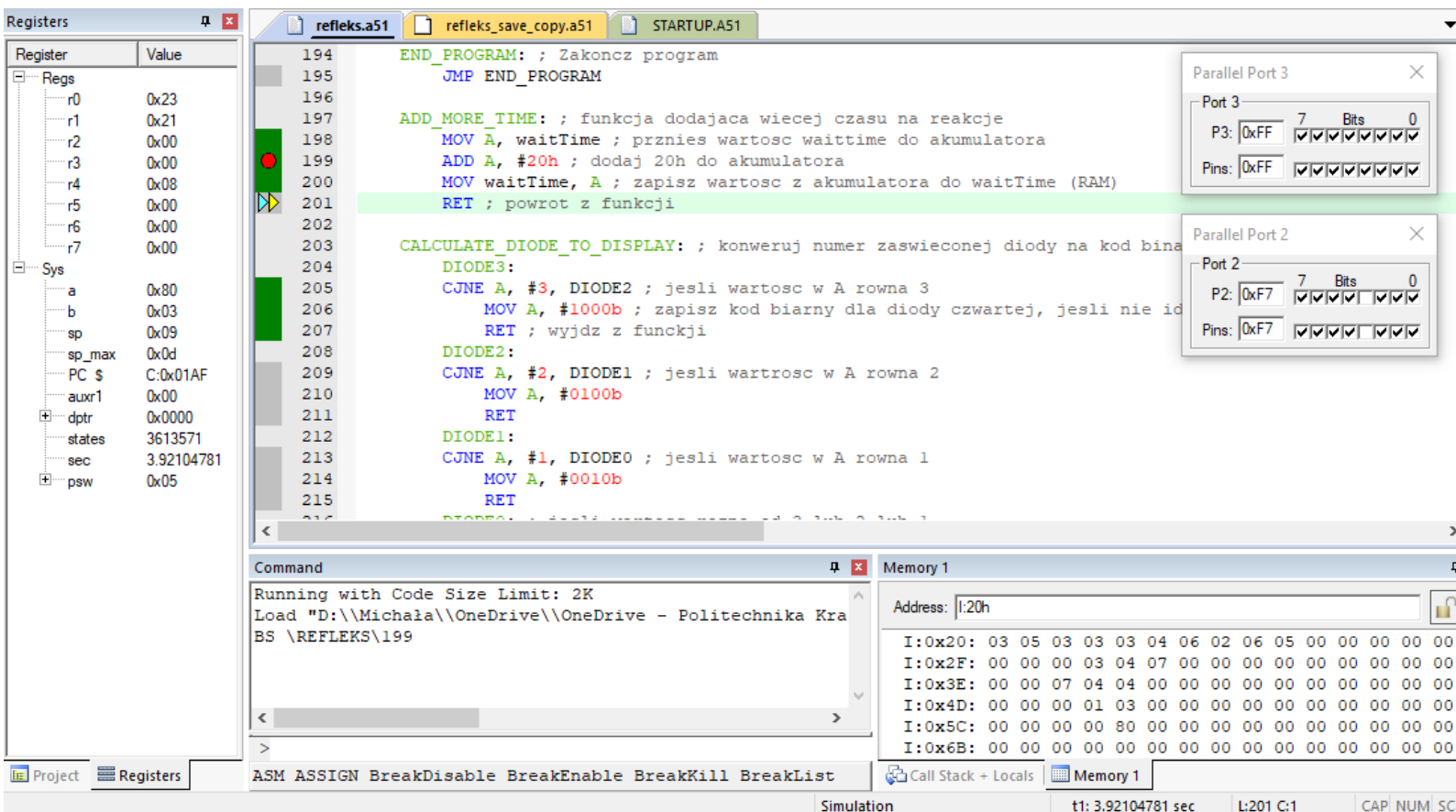
Rys. 7. Funkcja oczekująca

- W następnym kroku program przechodzi do etykiety **CHECK_ANSWER**, która wykonuje sprawdzenie, czy wciśnięty klawisz odpowiada wylosowanej diodzie. Jeśli tak, program idzie do funkcji **WIN**, która zapala sekwencję zwycięstwa na diodach (010101010), jeśli nie, idzie do etykiety **WRONG_ANSWER**;

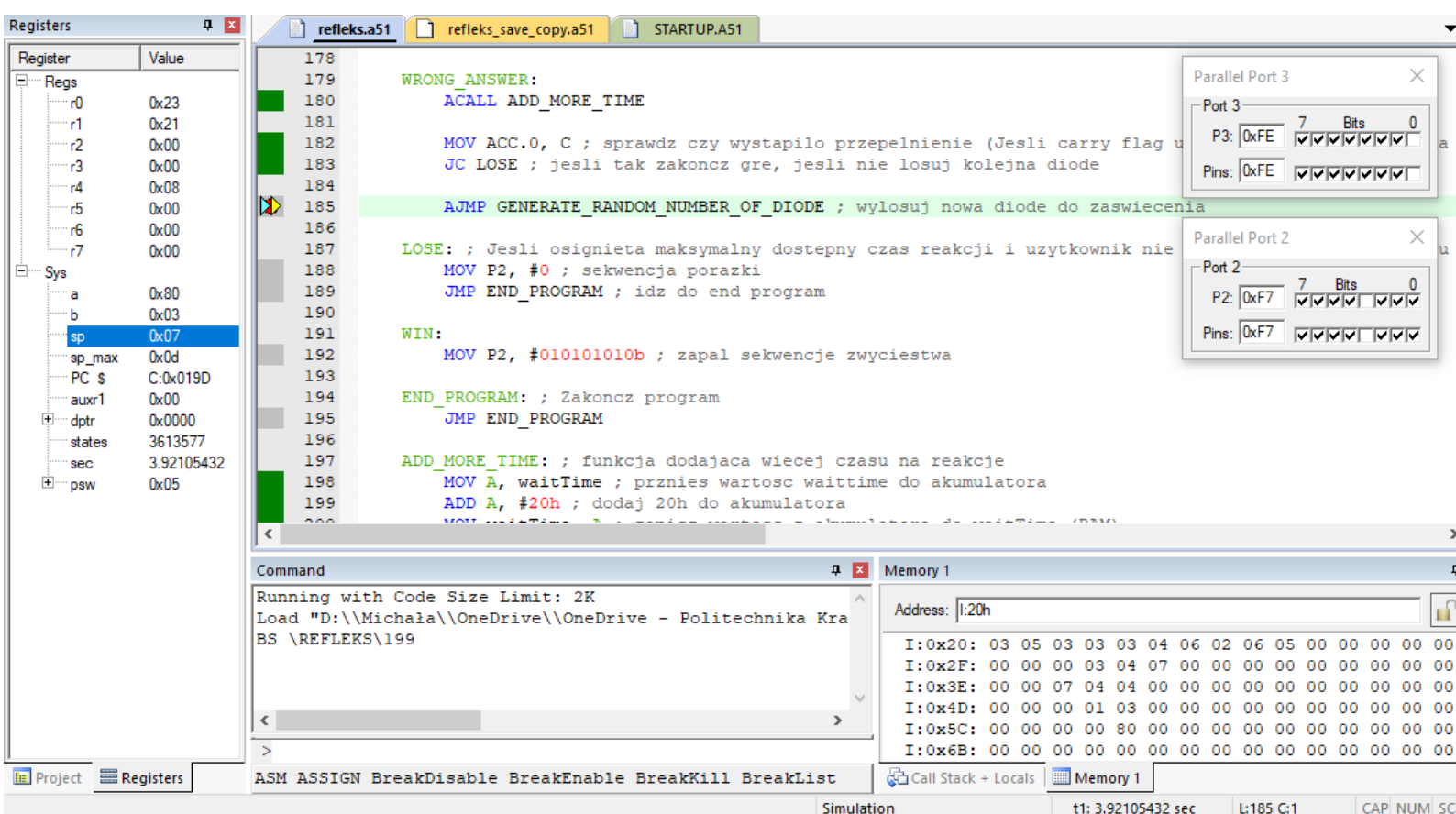


Rys. 8. Sprawdzanie poprawności odpowiedzi od użytkownika

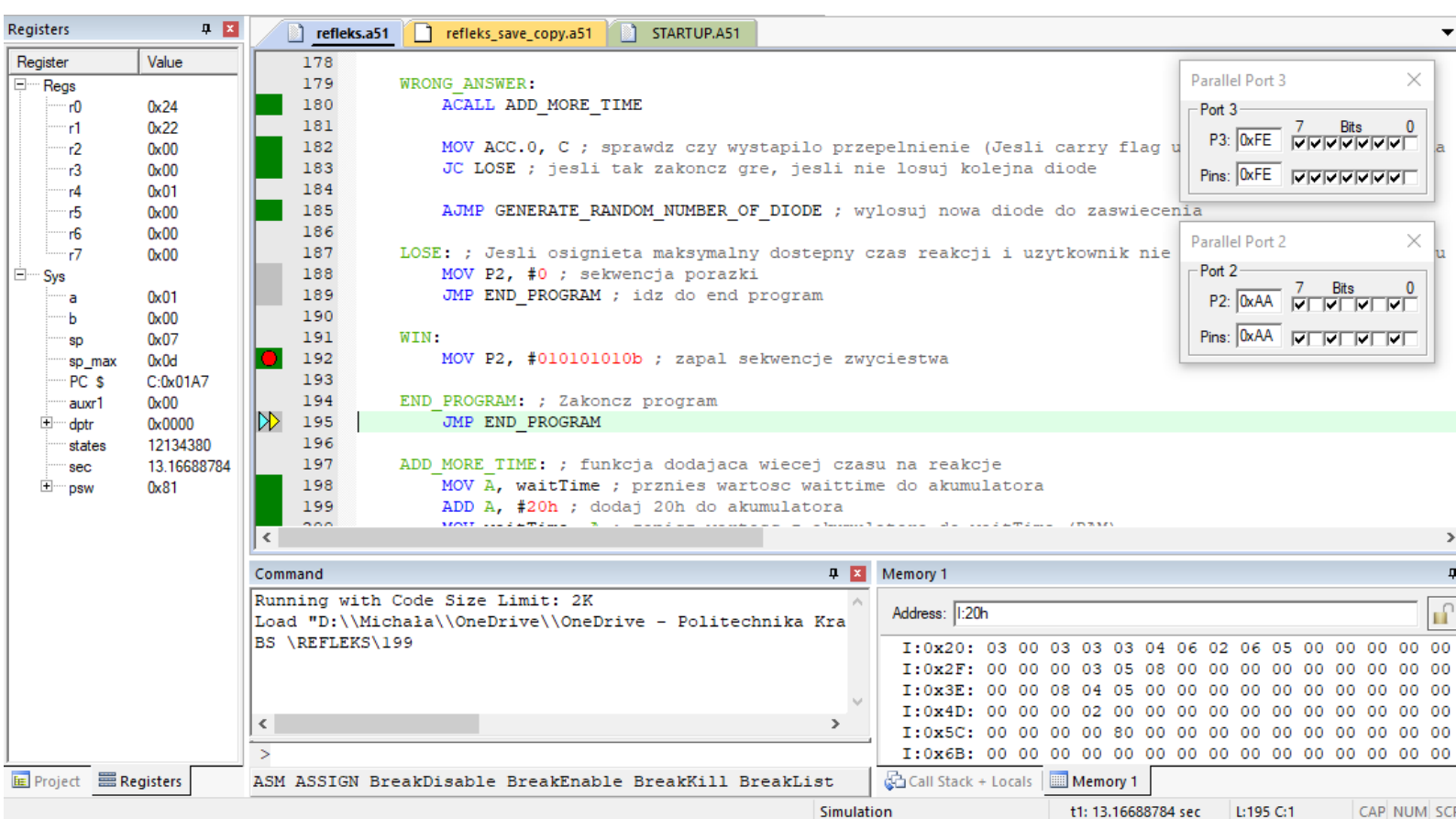
8. Jeśli udzielono błędnej odpowiedzi, wywoływana jest funkcja **ADD_MORE_TIME**, która dodaje 20h do czasu reakcji. Następnie sprawdzany jest bit przepełnienia, jeśli wystąpiło przepełnienie, oznacza to, że użytkownik zużył cały dostępny czas i program idzie do etykiety **LOSE**, gdzie wyświetlana jest sekwencja porażki (zapalone wszystkie diody) i program kończy działanie. Jeśli nie, gra toczy się dalej i jest generowana nowa dioda do zapalenia.



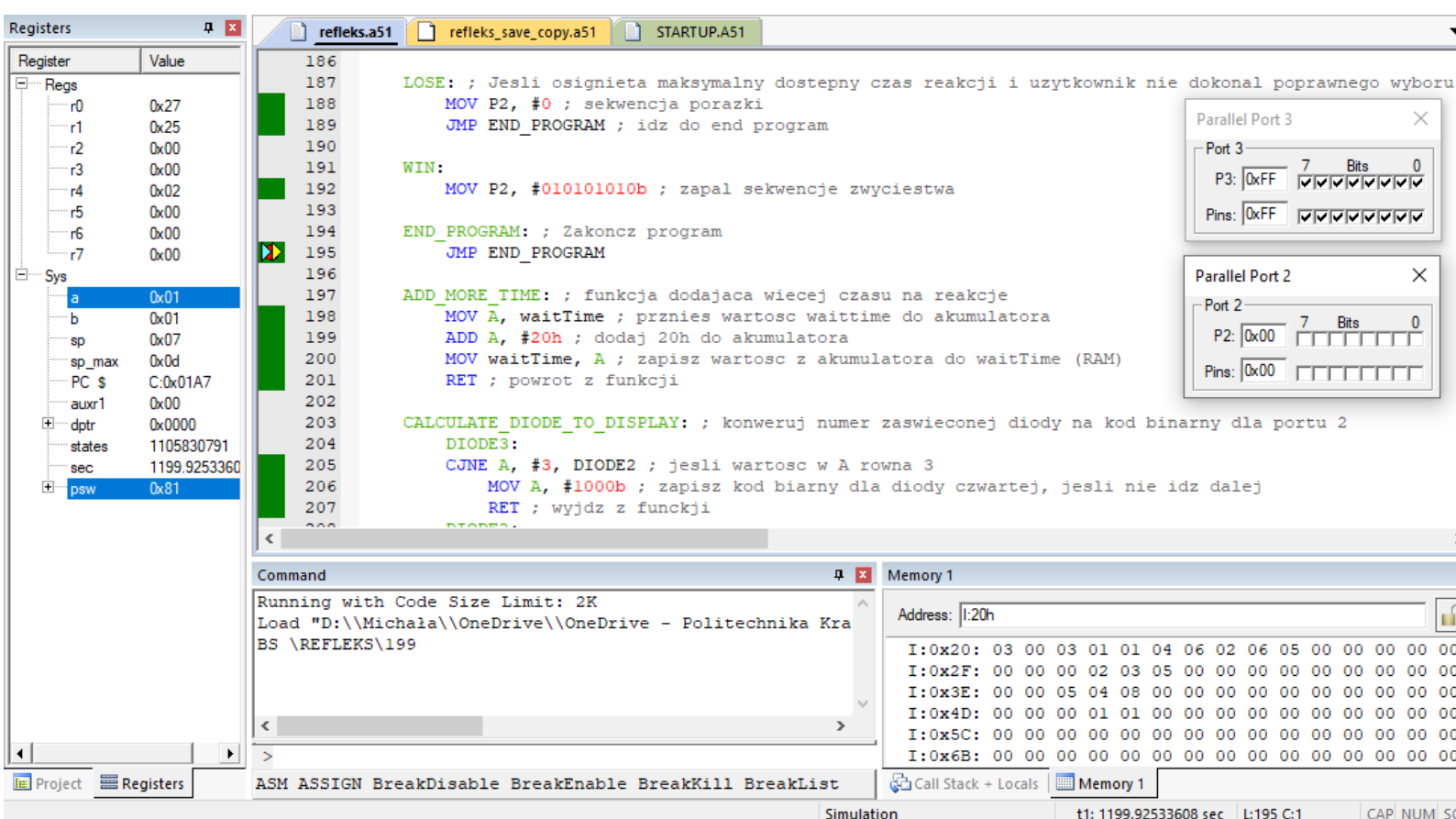
Rys. 10. Funkcja służąca do dodawania czasu dla reakcji użytkownika



Rys. 11. Funkcja służąca do sprawdzania warunku przegranej



Rys. 12. Wygrana



Rys. 13. Przegląd

6. Podsumowanie i wnioski:

Drugie ćwiczenie miało za zadanie przybliżyć nam obsługę portów wejścia / wyjścia w procesorze **Intel 8051** i obsługi prostych urządzeń podłączonych do wyprowadzeń mikrokontrolera, takich jak LED i przełączniki. Ponieważ porty są mapowane na wewnętrzną pamięć RAM (obszar rejestrów specjalnych), odczyt i zapis do portów odbywa się tak samo jak odczyt i zapis innych komórek pamięci.