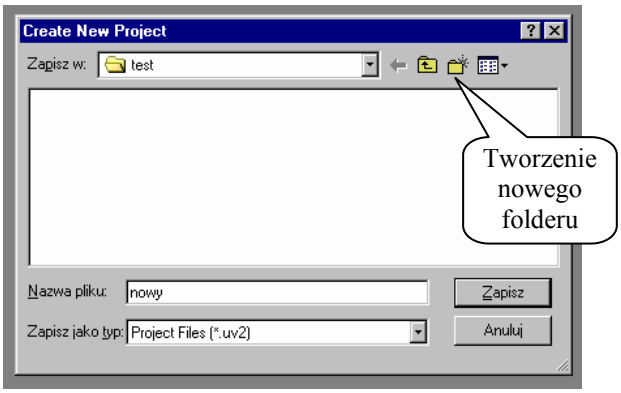
	LABORATORIUM ***PODSTAWY TECHNIKI MIKROPROCESOROWEJ***
L I	***Środowisko projektowe***

I. ZAPOZNANIE ZE ŚRODOWISKIEM PROJEKTOWYM µVISION

Środowisko projektowe µVISION zostało opracowane przez firmę KEIL SOFTWARE w celu wsparcia procesu tworzenia oprogramowania na mikroprocesory rodziny ’51. W skład pakietu wchodzą następujące programy narzędziowe:

- menadżer projektów − umożliwia tworzenie i zarządzanie projektem użytkownika;
- edytor tekstu − umożliwia wprowadzenie kodu źródłowego w języku asemblera lub C (interaktywna korekcja błędów pozwala na natychmiastowe sprawdzenie poprawności składniowej pisanego programu) oraz edycję plików wynikowych powstających w czasie tworzenia projektu;
- system pomocy − pozwala na uzyskanie natychmiastowej podpowiedzi;
- kompilator/asembler − tworzy plik obiektowy, sprawdza poprawność składniową programu;
- menadżer bibliotek − tworzy biblioteki programów (modułów), które mogą być w przyszłości wielokrotnie używane do tworzenia nowych programów użytkowych;
- linker − na podstawie plików obiektowych tworzy całkowity moduł obiektowy zawierający kompletny kod maszynowy programu użytkownika; kod ten może zostać wykorzystany w procesie programowania pamięci np. EPROM);
- konwerter formatów − zapisuje plik wynikowy z procesu linkowania w innym formacie (np. szesnastkowym *hex.*, zrozumiałym dla większości programatorów);
- symulator − w sposób programowy (wykorzystując wyłącznie możliwości komputera PC) symuluje zasoby mikroprocesora rodziny ’51;
- debugger − pozwala na uruchomienie programu w systemie z mikroprocesorem ADuC 845;
- monitor − program rezydentny umieszczony w mikroprocesorze ADuC 845; pozwala na nawiązanie komunikacji w standardzie RS232, pomiędzy debugger’em na komputerze PC a ADuC 845.



II. TWORZENIE PROJEKTU

1. Uruchomienie programu µVISION

Uruchom program µVISION dwukrotnie klikając w ikonę skrótu 

umieszczoną na pulpicie.

2. Tworzenie nowego projektu

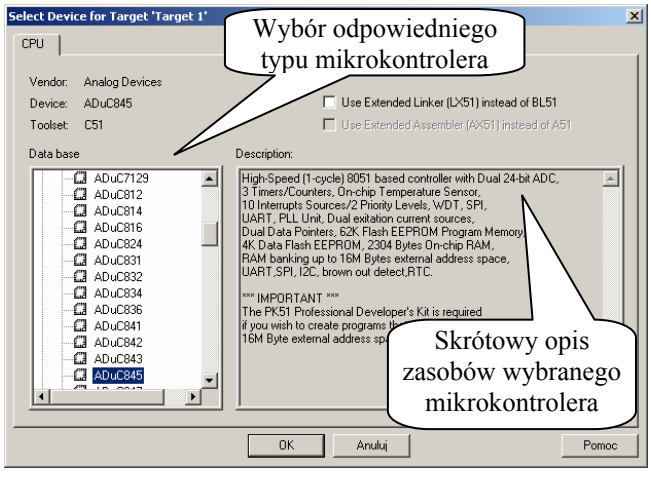
Z menu *Project* wybierz opcję *Close Project...* w celu zamknięcia domyślnego projektu.

Z menu *Project* wybierz opcję *New Project...* W katalogu *d:\Student\Lxx* utwórz projekt o nazwie *Nowy* .

Uwaga

Lxx − oznacza numer grupy laboratoryjnej.

Każdy nowy projekt powinien być utworzony w oddzielnym katalogu.



3. Wybór mikrokontrolera

W okienku *Select Device for Target 'Target 1'* wybierz odpowiedni typ mikrokontrolera:
Analog Devices / ADuC 845 .

Uwaga

Okienko *Select Device for Target 'Target 1'* jest również dostępne z menu *Project* .

Nie ma potrzeby dołączania do projektu „*Analog Devices Startup Code*".

4. Tworzenie pliku tekstowego

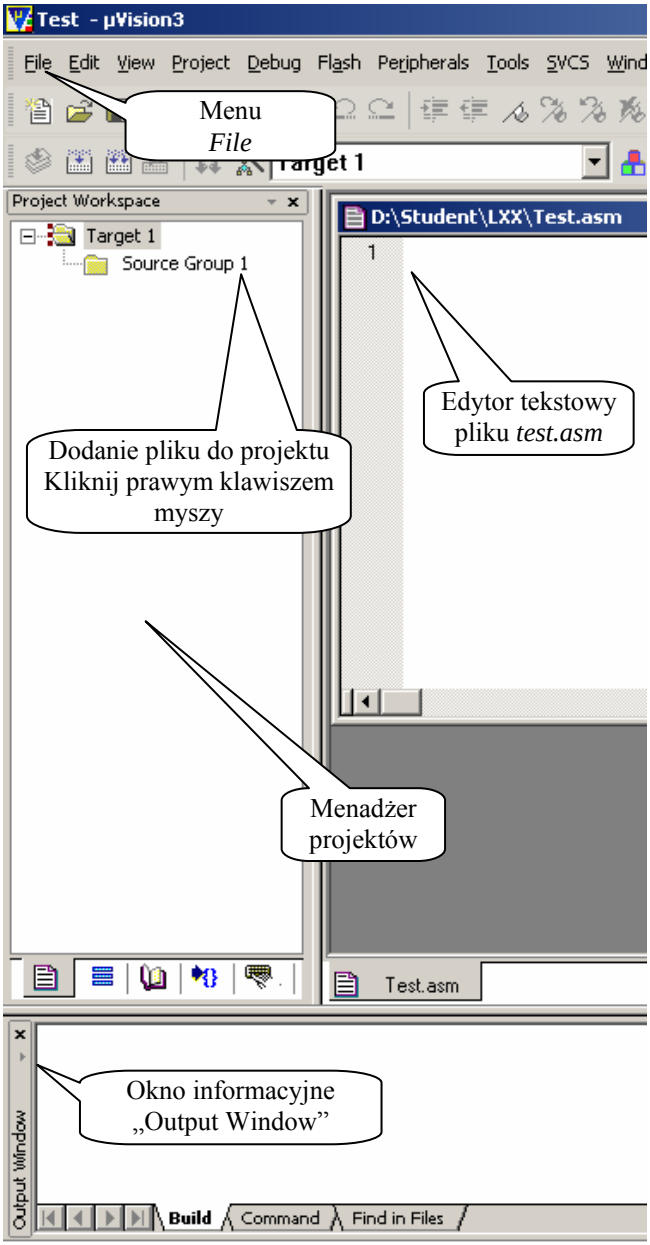
Korzystając z menu *File/New...* otwórz okienko edytora plików źródłowych, a następnie przy pomocy menu *File/Save As...* utwórz nowy plik o nazwie *test.asm* .

5. Dodanie pliku tekstowego do projektu

Klikając prawym klawiszem myszy na folderze *Source Group 1* w okienku menadżera projektów rozwiń lokalne menu i wybierz opcję *Add Files to Group 'Source Group 1'*; zlokalizuj plik *test.asm* i dodaj go do projektu.

6. Wprowadzenie kodu przykładowego programu

W okienku edytora tekstowego wprowadź poniższy tekst. Jest to kod źródłowy programu, którego zadaniem jest generowanie fali prostokątnej na wyprowadzeniu P2.0 z częstotliwością 10Hz. W ten sposób zostanie utworzony plik źródłowy o nazwie *test.asm*.



Uwaga

Nie zapomnij zapisać na dysku swojej pracy.

```
;*******************************************************************************
NAME generator                      ;dyrektywa=> deklaracja nazwy programu
                                    ;DEKLARACJE STAŁYCH=> dyrektywa asemblera EQU
T0_r            EQU   6666h         ;deklaracja stałej 16 bitowej
IRAM_r          EQU   07Fh          ;deklaracja stałej 8 bitowej
LED_cz          EQU   P3.3          ;deklaracja nowej nazwy bitu
LED_z           EQU   P3.5          ;deklaracja nowej nazwy bitu

                                    ;POCZĄTEK PROGRAMU=> dyrektywa asemblera CSEG
CSEG at 0000h                       ;początek segmentu kodu=> od adresu 0000h
     ljmp  Start                    ;rozkaz=> skocz do etykiety Start
                                    ;POCZĄTEK OBSŁUGI PRZERWANIA OD T0
ORG 000Bh                           ;dyrektywa=> umieść kod od adresu 000Bh
     ljmp  Int_T0                   ;rozkaz=> skocz do etykiety Int_T0

                                    ;INICJALIZACJA MIKROPROCESORA
ORG 0100h                           ;dyrektywa=> umieść kod od wskazanego adresu
Start:                              ;etykieta=> wskazuje miejsce w programie
     mov   SP,#60                   ;rozkaz=> wpisz do rejestru SP liczbę #60
     mov   R0,#IRAM_r               ;rozkaz=> wpisz do rejestru R0 liczbę #IRAM_r
```

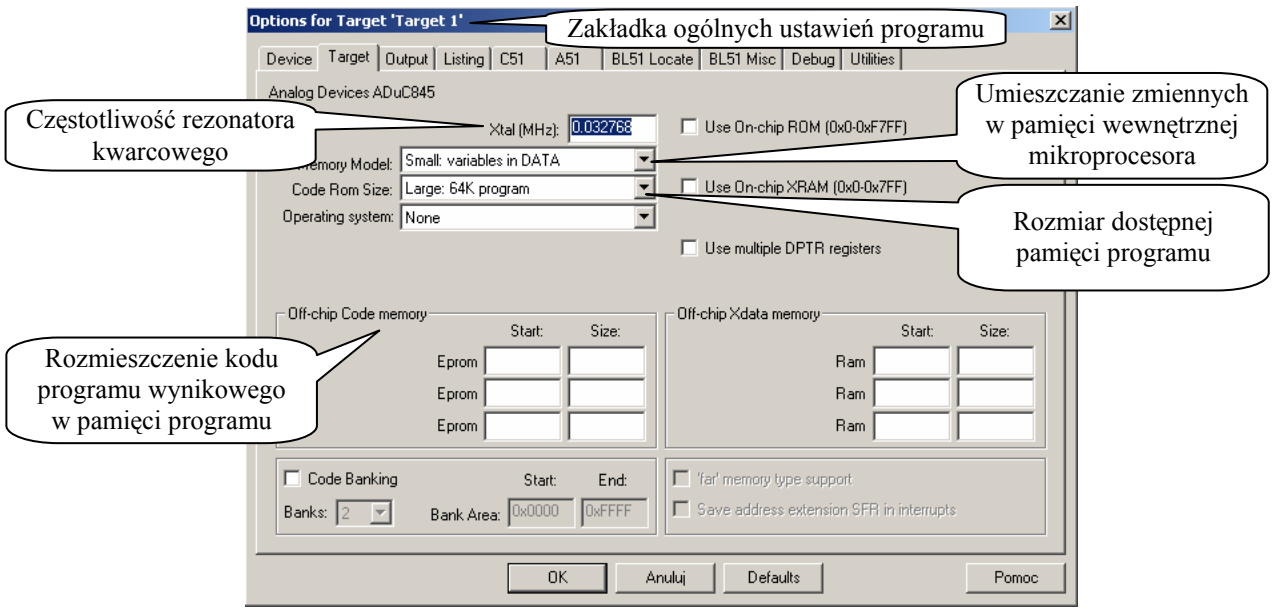
```
      clr   A                     ;rozkaz=>  wyczyść akumulator

Reset:
      mov   @R0,A                 ;@ => adresowanie pośrednie zawartością r. R0
      djnz  R0,Reset              ;rozkaz=> skok warunkowy:
                                  ; R0 = R0 - 1
                                  ; jeżeli R0 = 0 - wykonaj następny rozkaz
                                  ; jeżeli R0 ≠ 0 - skocz do etykiety Reset
      orl   IE,#1000$0010b        ;rozkaz=> ustaw w IE wskazane jedynkami bity
      orl   TMOD,#0000$0001b      ;rozkaz=> ustaw w TMOD bity
      mov   TH0,#HIGH(T0_r)       ;wpisz do TH0 starszy HIGH bajt liczby T0_r
      mov   TL0,#LOW(T0_r)        ;wpisz do TL0 młodszy LOW bajt liczby T0_r
      setb  TR0                   ;rozkaz=> ustaw bit TR0
      mov   P2,#00                ;rozkaz=> wpisz do rejestru P2 liczbę #00
      clr   LED_cz                ;rozkaz=> wyzeruj bit P3.3
      clr   LED_z                 ;rozkaz=> wyzeruj bit P3.5
Petla:                            ;PĘTLA GŁÓWNA PROGRAMU
      cjne  R1,#25,Petla          ;rozkaz=> skok warunkowy:
                                  ; jeżeli R1 = 25 - wykonaj następny rozkaz
                                  ; jeżeli R1 ≠ 25 - skocz do etykiety Petla
      mov   R1,#0                 ;rozkaz=> wyzeruj rejestr R1
      cpl   P2.0                  ;rozkaz=> zaneguj bit P2.0
      jmp   Petla                 ;rozkaz=> skocz do etykiety Petla

Int_T0:                           ;OBSŁUGA PRZERWANIA OD LICZNIKA T0
      mov   TH0,#HIGH(T0_r)       ;wpisz do TH0 starszy HIGH bajt liczby T0_r
      mov   TL0,#LOW(T0_r)        ;wpisz do TL0 młodszy LOW bajt liczby T0_r
      inc   R1                    ;rozkaz=> zwiększ o 1 rejestr R1
      reti                        ;rozkaz=> koniec przerwania
END                               ;dyrektywa=> koniec programu
;*******************************************************************************
```

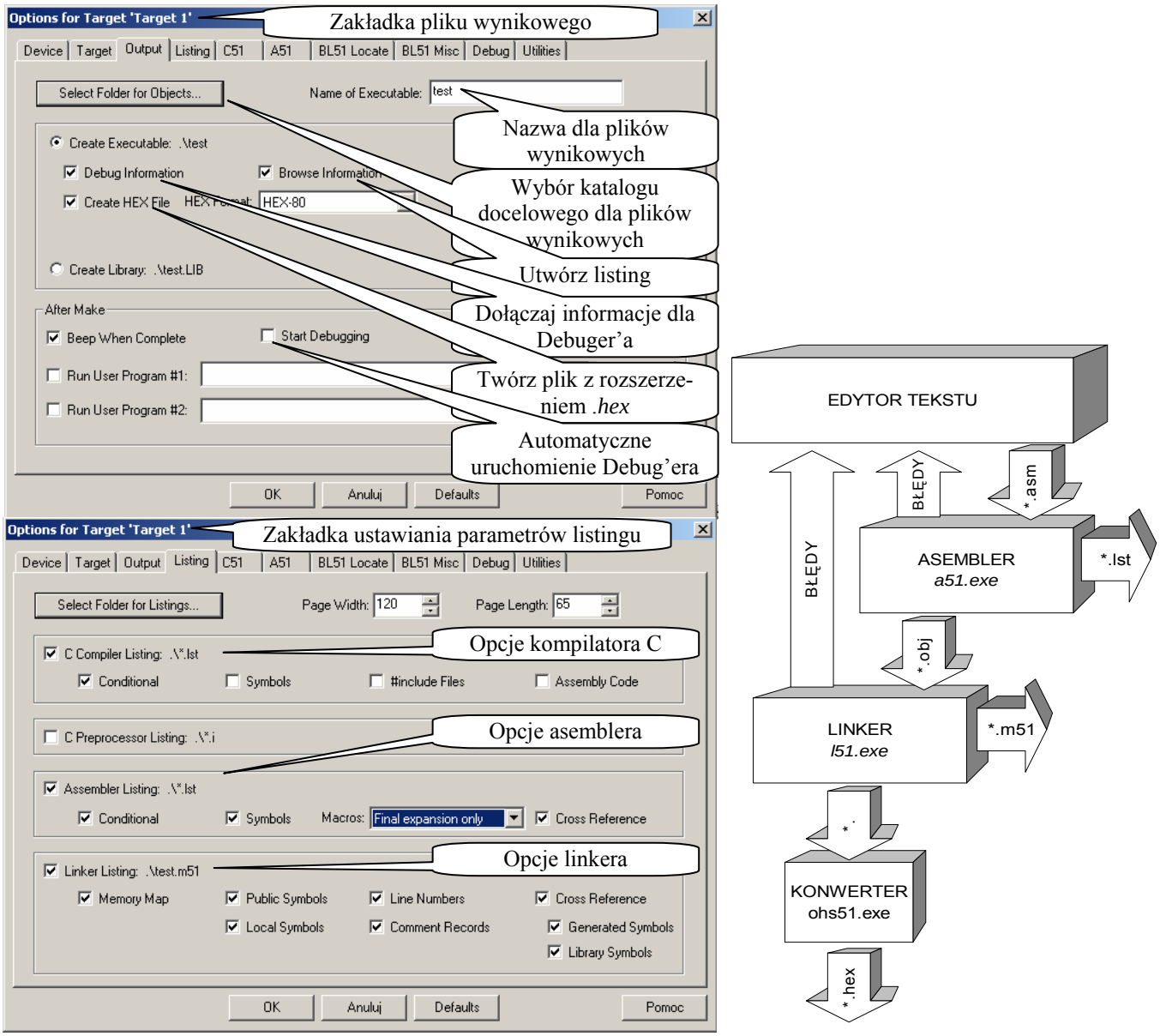
7. Ustawienie parametrów przetwarzania utworzonego zbioru źródłowego

Klikając prawym klawiszem myszy na folderze *Target 1* w okienku menadżera projektów rozwiń lokalne menu i wybierz opcję *Options for Target 'Target 1'*. Ustaw odpowiednio parametry przetwarzania pliku źródłowego (zgodnie z poniższymi rysunkami).

Uwaga

W katalogu *d:\Student\Lxx* sprawdź, jakie pliki zostały utworzone do tej pory – zanotuj ich rozszerzenia.





8. Budowanie projektu i tworzenie pliku z kodem wynikowym

Korzystając z menu *Project/Build Target* (lub *Project/Rebuild all Target Files*) można uruchomić proces kompilacji i tworzenia pliku wynikowego. Ewentualne informacje o błędach (ang. error) i ostrzeżeniach (ang. warning) zostaną wyświetlone w dolnym okienku informacyjnym „*Output Window*".

Proces kompilacji wykorzystuje trzy programy:

- asembler − przetwarza plik z kodem źródłowym *test.asm*; w wyniku zwraca plik obiektowy z rozszerzeniem *.obj* oraz plik informacyjny (o przebiegu procesu asemblacji) z rozszerzeniem *.lst*;
- linker − przetwarza plik obiektowy z rozszerzeniem *.obj* lub łączy kilka plików obiektowych z plikami bibliotecznymi; w wyniku zwraca plik z kodem maszynowym bez rozszerzenia oraz plik informacyjny (o przebiegu procesu linkowania) z rozszerzeniem *.m51*.
- konwerter − przetwarza plik z kodem maszynowym bez rozszerzenia na plik z rozszerzeniem *.hex,* zawierający kod maszynowy w formacie Intel−hex, zrozumiały dla większości programatorów.

Uwaga

Pliki wynikowe można przeglądać w edytorze po otwarciu ich za pośrednictwem menu: *File/Open.*

Project/Build Target − kompilowanie tylko tych plików, które zostały zmienione po ostatnim procesie kompilacji.

Project/Rebuild all target files − ponowne przekompilowanie wszystkich plików.

Project/Translate d:\Student\Lxx\test.asm – przeprowadzenie procesu asemblacji.

III. DODATKOWE ZAGADNIENIA

A. Sprawdź efekty wprowadzenia zmian / błędów w kodzie źródłowym (patrz tabela w p. IV. 3)

B. Zlokalizuj poniższe elementy w pliku wynikowym procesu asemblacji:
- datę utworzenia zbioru,
- użyte dyrektywy asemblera,
- informację o błędach,
- numer użytego banku rejestrów,
- listing programu (adres rozkazu w segmencie, kod rozkazu, znacznik F (argument rozkazu musi zostać wyliczony przez linker), numer linii w listingu),
- tablicę symboli: pole NAME (nazwy użytych symboli), pole TYPE (typ segmentu: C – kodu, D – DATA, I – IDATA, B – bit, N – stała), pole VALUE (wartość liczbowa symbolu), pole ATTRIBUTES/REFERENCES (atrybuty i miejsce użycia symbolu, # – miejsce definicji symbolu).

C. Wprowadź zmiany w zakładce *Listing* okienka *Options for Target 'Target 1'* (punkt II. 7) – zwróć szczególną uwagę na pole *Asembler Listing:* oraz *Page Width* i *Page Length*.

D. Zlokalizuj następujące elementy w pliku wynikowym procesu linkowania:
- tablicę rozmieszczenia segmentów (adres początkowy, końcowy, wielkość)
- tablicę symboli: pola NAME, TYP, VALUE (patrz punkt III. 2),
- informację o błędach,
- datę utworzenia zbioru,
- uzyte dyrektywy linkera.

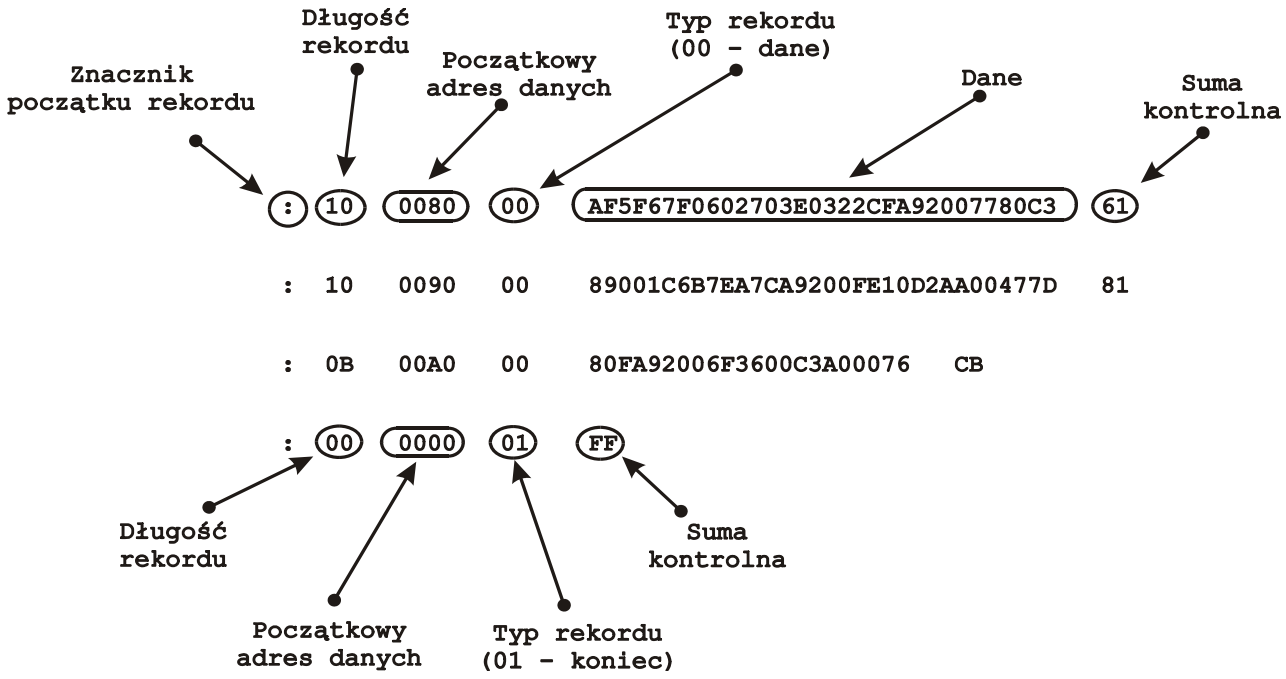
E. Wprowadź zmiany w zakładce *Listing* okienka *Options for Target 'Target 1'* (punkt II. 7) – zwróć szczególną uwagę na pole *Linker Listing:*.

F. Zapoznaj się z zawartością zbioru wynikowego z rozszerzeniem *.hex*.

Format zbioru INTEL–HEX został opracowany przez firmę INTEL w celu standaryzacji postaci zbiorów wynikowych. Jest on używany dla zbiorów o wielkości do 64kB, dlatego też najczęściej stosowany jest przez narzędzia i sterowniki wykorzystujące mikroprocesory 8–bitowe. Wszystkie dane są zapisywane w plikach tekstowych w postaci szesnastkowej przy użyciu znaków ASCII: „0" ... „9" oraz „A" ... „F". Dane są pogrupowane w rekordy (linie), które zawierają także dodatkowe informacje.

Typowa linia z danymi ma następującą postać:
- 1 znak => dwukropek – jest to nagłówek rekordu,
- 2, 3 znak => liczba **x** – bajtów danych zawartych w rekordzie,
- 4, 5, 6, 7 znak => 16 bitowy adres, pod który mają zostać zapisane dane z rekordu,
- 8, 9 znak => typ rekordu – w tym przypadku zawsze 00,
- 10, ..., y znak => dane (np. rozkazy i ich argumenty); y = x * 2 + 9,
- y + 1, y + 2 znak => suma kontrolna,
- y + 3, y + 4 znak => znaki CR + LF (koniec linii + powrót karetki).
- Ponadto format INTEL–HEX definiuje rekord końcowy w postaci:
- 1 znak => dwukropek – jest to nagłówek rekordu,
- 2, 3 znak => liczba x = 00 – bajtów danych zawartych w rekordzie,
- 4, 5, 6, 7 znak => adres 0000,
- 8, 9 znak => typ rekordu – w tym przypadku zawsze 01,
- 10, 11 znak => suma kontrolna,
- 12, 13 znak => znaki CR + LF (koniec linii + powrót karetki).

Suma kontrolna jest dopełnieniem (kod U2) 8–bitowej sumy (bez przeniesienia) wszystkich bajtów w rekordzie. Dla rekordu końcowego suma wynosi „FF" w zapisie szesnastkowym. Postać zbioru wynikowego *test.hex* prezentuje poniższy rysunek.



Zbiór wynikowy w standardzie INTEL–HEX.

IV. PRZYGOTOWANIE DO NASTĘPNYCH ZAJĘĆ

1. Wiedza teoretyczna

A. Blokowa struktura mikroprocesora jednoukładowego (przeznaczenie poszczególnych bloków).

2. Wiadomości z ćwiczenia pierwszego

A. Proces tworzenia programu.

B. Znaczenie poznanych opcji asemblera i linkera.

C. Znaczenie informacji zawartych w zbiorach wynikowych z procesu asemblacji i linkowania.

D. Struktura i znaczenie rozkazów użytych w przykładowym programie.

3. Spróbuj utworzyć poniższe tabele

Rodzaj wprowadzonej błędnej poprawki	Komunikat o błędzie odczytany z dolnego okienka „Output Window” lub z pliku *test.lst*

Zmiany (błędy) wprowadź w różnym miejscu pliku źródłowego. Mogą one polegać np. na:

- – skasowaniu jednej linii programu,
- – skasowaniu etykiety,
- – wprowadzeniu błędnych nazw rozkazu lub dyrektywy,
- – użyciu błędnej składni rozkazu lub dyrektywy,
- – wprowadzeniu zbyt dużej liczby,
- – wprowadzeniu bardzo dużej liczby znaków (np. 300) w jednej linii.

Opcja asemblera lub linkera	Sposób wywołania	Zauważone różnice