AUTOMATYKA • 2005 • Tom 9 • Zeszyt 3

Jakub Janiak*, Zbigniew A. Nowacki**

Zintegrowane środowisko programowania niskopoziomowego ASMEdit

1. Wprowadzenie

Trudno nie zgodzić się ze stwierdzeniem, iż „oprogramowanie jest niewątpliwie najciekawszym produktem inżynierskim, jaki kiedykolwiek wynaleziono" [1]. Często zdarza się jednak tak, że nie spełnia ono wymagań, jakie nań położono, gdyż działa wadliwie. Najczęstszym tego powodem jest utrata kontroli intelektualnej nad projektem.

Programiści piszący kod w nowoczesnych językach wysokopoziomowych mają ułatwione zadanie, gdyż bardzo wiele pomocy otrzymują od środowiska programistycznego, w którym tworzą daną implementację. Niestety, tego udogodnienia są pozbawieni programiści używający języka asemblera. Jako przykład można podać kompilator dsm51ass firmy MicroMade [3], xa51 napisany w Instytucie Automatyki Politechniki Łódzkiej [4], a także kompilator dla procesorów rodziny Intel 8086-tasm [5, 8]. W ich przypadku stosowanie podstawowych zasad programowania, takich jak: „dziel i rządź" czy „wykorzystaj poprzednie prace" nie jest wspierane przez środowisko programistyczne. Oczywiste jest, iż wraz ze wzrostem rozmiaru projektu utrudnione staje się wyszukiwanie błędów oraz pielęgnacja kodu. W szczególności problem ten dotyka tworzenia aplikacji dla mikrokontrolerów jednoukładowych, używanych powszechnie np. w urządzeniach przenośnych.

Aby poradzić sobie z zaistniałym problemem programista asemblerowy ma do wyboru podejście klasyczne, tzn. może opierać się głównie na swoim doświadczeniu, lub może posłużyć się aplikacją wspomagającą jego pracę.

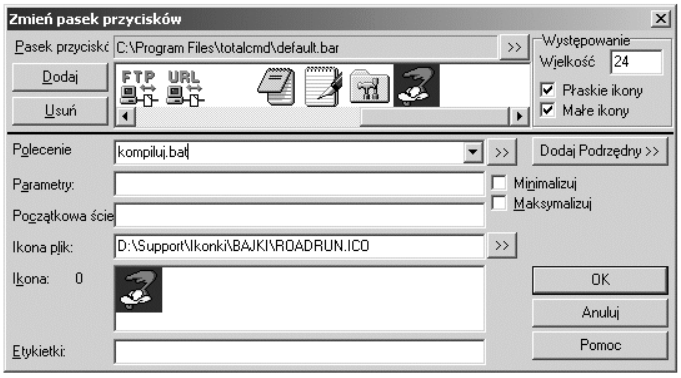
2. Podejście klasyczne

Klasyczne podejście do powyższego zagadnienia sprowadza się do wykonania trzech czynności:

1) utworzenia kodu programu w pliku tekstowym,
2) kompilacji,
3) uruchomienia kodu na dedykowanej platformie sprzętowej.

* Instytut Obrabiarek i Technologii Budowy Maszyn, Politechnika Łódzka
** Katedra Informatyki Stosowanej, Politechnika Łódzka

Przy założeniu, że kod jest poprawny, wymaga to wydania przynajmniej trzech sparametryzowanych komend z linii poleceń. Oczywiście powyższą procedurę można natychmiast usprawnić, stosując pliki wsadowe. Podobnie można postąpić w przypadku, jeśli zachodzi konieczność debugowania programu. Widać jednak, iż liczba czynności do wykonania, nawet w przypadku optymistycznym (bez błędów), jest duża. Aby polepszyć komfort pracy, można wykorzystać gotowe oprogramowanie, np. menedżer plików Total Commander (dostępny na: www.ghisler.com). Należy wówczas utworzyć własny pasek narzędzi i podłączyć do wygenerowanych przycisków akcje – wywołania odpowiednich plików wsadowych, co pokazano na rysunku 1.



Rys. 1. Tworzenie i zmiana paska narzędzi w aplikacji Total Commander

Działając konsekwentnie i uważnie, uzyska się uproszczenie i znaczne przyspieszenie zarówno procesu kompilacji, jak i uruchamiania aplikacji. W kwestii stylu projektowania i programowania twórca jest nadal zdany na siebie, bazuje tylko i wyłącznie na swojej wiedzy i doświadczeniu. Jest to niewątpliwie wada podejścia klasycznego, szczególnie dla początkujących programistów.

3. Aplikacja ASMEdit

Rozwiązaniem powyższego problemu jest napisana przez autorów na platformie Microsoft .NET, oparta na architekturze *dokument*/*widok* i przeznaczona dla mikroprocesorów rodziny 8051 wielodokumentowa aplikacja ASMEdit [2]. Pomysł stworzenia środowiska programistycznego dla języka asemblera powstał z powodu braku takiego systemu na licencji *freeware* nawet wśród oprogramowania komercyjnego. W szczególności problem ten dotyka programowania mikrokontrolerów jednoukładowych rodziny np. ‘51. Jakkolwiek istnieją środowiska dla kompilatorów języka C/C++ (np. Keil Electronics), to do programowania niskopoziomowego dla jakiegokolwiek asemblera są one trudno osiągalne [2].

Ponieważ widok okna roboczego dla dokumentu ASMEdit został oparty na klasie CRichEditView będącej klasą *Microsoft Foundation Class* (*MFC*) [6, 7], daje to w konse-

kwencji niezbędne funkcje edycyjne. Moduł odpowiedzialny za kompilację projektu wykorzystuje kompilator skrośny xa51 autorstwa mgr inż. L. Pobiegi z Instytutu Automatyki Politechniki Łódzkiej. Celem rozszerzenia możliwości systemu, zastosowano darmowe oprogramowanie zewnętrzne, zintegrowane z aplikacją główną za pomocą plików wsadowych. Całość uzupełnia plik pomocy, zwierający szczegółową instrukcję obsługi.

Wraz z uruchomieniem ASMEdit, ładowane są dwa pliki konfiguracyjne:

1) plik konfiguracyjny aplikacji głównej (appset.set),
2) plik konfiguracyjny (*.cfg) modułu sprawdzania pisowni rozkazów i operandów (parsera).

Opcjonalnie – jeśli zostało to uwzględnione w pliku konfiguracyjnym aplikacji głównej – ładowany jest również szablon projektu (*.tpl) oraz paczki procedur (*.pkg). Pliki cfg, tpl oraz pkg zostały zaprojektowane na podstawie opracowania [9]. Poniżej zostaną krótko omówione poszczególne typy plików.

Plik konfiguracyjny aplikacji głównej przechowuje informacje o ścieżkach dostępu do oprogramowania zewnętrznego oraz o ustawieniach graficznego interfejsu użytkownika GUI (*graphic user interface*). Ustawienia te mogą zostać zmienione odpowiednio w oknach dialogowych: „Ustawienia główne aplikacji" oraz „Ustawienia ścieżek dostępu".

Plik konfiguracyjny parsera zawiera informacje konieczne do poprawnego działania modułu pomocy ASMEdit, tzn. mnemoniki rozkazów, operandy oraz informacje pomocnicze. Jest zatem oczywiste, iż jest on unikalny dla danego asemblera. Dodano również szablony nagłówków programu głównego i podprogramów.

Szablon projektu to plik tekstowy, w którym znajduje się wzorzec typowego programu oraz znaczniki pozwalające na szybkie przenoszenie się pomiędzy blokami kodu. Samo wydzielenie bloków nadaje aplikacji znacznie bardziej czytelną postać.

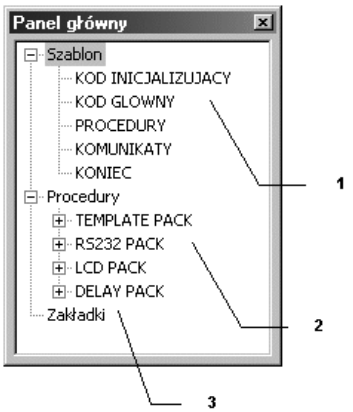
Ostatni typ plików to paczki procedur. Są to pliki zawierające podprogramy przeznaczone do realizacji specjalistycznych zadań. Standardowy pakiet pozwala na realizację transmisji szeregowej łączem RS-232, obsługę wyświetlacza LCD opartego na kontrolerze HD 44780 i dodanie pętli opóźniających.

Środowisko ASMEdit komunikuje się z użytkownikiem-programistą za pomocą graficznego interfejsu użytkownika. Jest on wzorowany na GUI, jaki posiada Microsoft Developement Studio 6.0, który zdaniem autorów wyróżnia się doskonałą funkcjonalnością oraz przejrzystością.

Podstawowe elementy tego interfejsu to:

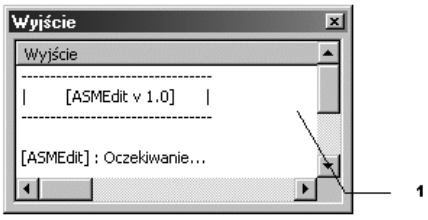
– dokowalne panele: główny, operandów oraz wyjściowy;
– dokowalne paski narzędzi;
– menu główne;
– menu kontekstowe (dostępne po włączeniu odpowiedniej opcji).

Panel główny (rys. 2) umożliwia szybkie przemieszczanie się po blokach kodu. Swoją reprezentację znajdują tu wspomniane wcześniej znaczniki szablonu projektu oraz predefiniowane procedury. Jeśli zaistnieje taka potrzeba, można również dodać własne znaczniki – tzw. zakładki (mają one takie same cechy jak znaczniki szablonu projektu).



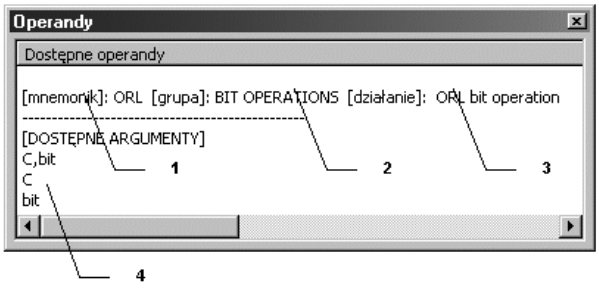
Rys. 2. Elementy panelu głównego: 1 – blok znaczników szablonu, 2 – blok procedur, 3 – blok zakładek

Panel wyjściowy (rys. 3) wyświetla aktualny stan aplikacji ASMEdit oraz wyniki ostatniej kompilacji wraz z ewentualnymi numerami linii, w których wykryto błąd.



Rys. 3. Elementy panelu wyjściowego: 1 – pole komunikatów

Panel operandów (rys. 4) wyświetla (jeśli opcja została włączona) pełne informacje o mnemoniku rozkazu, który został wybrany poprzez kliknięcie prawym klawiszem myszy w obszarze roboczym.



Rys. 4. Elementy panelu operandów: 1 – aktualnie wybrany mnemonik, 2 – grupa do której przypisany został mnemonik, 3 – geneza skrótu lub jego opis działania, 4 – lista dostępnych argumentów

Dodatkowo istnieje możliwość uzyskania skróconej wersji tej informacji poprzez menu wywołanie kontekstowego (również ta opcja musi zostać włączona), co ilustruje rysunek 5.

Rys. 5. Menu kontekstowe: 1 – aktualne położenie kursora oraz całkowita liczba linii, 2 – aktualnie przetwarzany mnemonik, 3 – krótki opis danej operacji

Paski narzędzi są ściśle związane z menu głównym aplikacji. Ich obsługa jest intuicyjna i nie zostanie szczegółowo omówiona w niniejszym artykule. Jedyny pasek, który nie posiada swojej reprezentacji w menu głównym, to pasek „Parsowanie" (wspomniana kontrola pisowni rozkazów oraz operandów), ponieważ zawiera przyciski konieczne do uruchomienia elementów podsystemu wspomagania projektowania opisanych wcześniej.

Aby zapewnić niezbędną funkcjonalność, autorzy dodali do środowiska również aplikacje zewnętrzne, w dużej mierze usprawniające proces projektowania programu niskopoziomowego. Są to m.in. debugger Emily52 (DOS), debugger 8051 (Windows), tablica kodów ASCII (DOS), monitor portów Portmon (Windows). Usprawnione zostało również uruchamianie wbudowanych narzędzi Windows, takich jak hyperterminal, kalkulator czy linia poleceń. Dokładny opis tego oprogramowania znajduje się w plikach pomocy.

Ponieważ ASMEdit może współpracować z zewnętrznym emulatorem/programatorem procesorów rodziny '51, zintegrowana została również niezbędna aplikacja współpracująca (WinAte-01) wraz z odpowiednim plikiem wsadowym do uruchamiania skompilowanego programu na dedykowanej platformie sprzętowej.

4. Wyniki testów

Testowanie aplikacji wykazało, iż spełnia ona postawione wymagania odnośnie do elastyczności oraz wsparcia programowania. Udało się również usunąć większość błędów (w tym błąd zawieszania się kompilatora xa51) oraz poprawić jej funkcjonalność. Warto dodać, że dzięki modułowej budowie szkielet programu pozostał ten sam, natomiast zmiany dokonywane były jedynie w obrębie odpowiednich klas.

5. Podsumowanie

Celem projektu było zaopatrzenie projektanta w niezbędne oprogramowanie potrzebne do diagnozowania błędów oraz usprawniające proces tworzenia systemu, wzorem środowiska Microsoft Visual C++. Mechanizmy MFC umożliwiły taką integrację; część użytych programów to wspomniane programy darmowe na licencji *freeware*, a druga część to programy wkomponowane w system Windows, których uruchamianie jedynie usprawniono.

W przyszłości istnieje możliwość wyposażenia ASMEdit w kolejne przydatne funkcje oraz „sukcesywne przygotowywanie pakietów konfiguracyjnych dla innych kompilatorów dowolnego języka asemblera” [2].

Literatura

[1] Hamlet D., Maybee J.: *Podstawy techniczne inżynierii oprogramowania*. Warszawa, WNT 2003
[2] Janiak J.: *Praktyczne zastosowanie zasad analizy, projektowania i programowania obiektowego.* Łódź, Politechnika Łódzka 2005 (Praca magisterska)
[3] Gałka P., Gałka P.: *Podstawy programowania mikrokontrolera 8051*. Warszawa, Mikom 2005
[4] Mroczek H.: *Podręcznik użytkownika systemu MKD-51*. Politechnika Łódzka, 1995
[5] Duntemann J.: *Zrozumieć asembler*. Warszawa, Translator 1993
[6] Bates J., Tim Tompkins T.: *Poznaj Visual C++*. Warszawa, Mikom 1999
[7] Chapman D.: *Visual C++ dla każdego*. Gliwice, Helion 1999
[8] Dudek A.: *Jak pisać wirusy.* Warszawa, ReadMe 1994
[9] Rydzewski A.: *Mikrokomputery jednoukładowe*. Warszawa, WNT 1994