Podziękowania (11)
Wstęp (13)
Rozdział 1. Myśl niskopoziomowo, koduj wysokopoziomowo (17)

- 1.1. Nieporozumienia dotyczące jakości kompilatorów (18)
- 1.2. Dlaczego nadal warto uczyć się asemblera? (18)
- 1.3. Czemu znajomość asemblera nie jest absolutnie niezbędna? (19)
- 1.4. Myśl niskopoziomowo (19)
    - 1.4.1. Kompilatory są tylko tak dobre, jak dobry kod do interpretacji otrzymują (20)
    - 1.4.2. Pomóżmy kompilatorowi generować lepszy kod maszynowy (20)
    - 1.4.3. Jak myśleć o asemblerze, pisząc programy w językach wysokiego poziomu? (21)
- 1.5. Pisanie kodu wysokopoziomowego (23)
- 1.6. Założenia (23)
- 1.7. Niezależność od konkretnego języka (24)
- 1.8. Cechy kodu doskonałego (24)
- 1.9. Wymagane środowisko (25)
- 1.10. Dodatkowe informacje (26)

Rozdział 2. A może warto poznać asemblera? (27)

- 2.1. Kłody rzucane pod nogi uczącym się asemblera (27)
- 2.2. Tom drugi Profesjonalnego programowania spieszy z odsieczą (28)
- 2.3. Wysokopoziomowe asemblery przychodzą z pomocą (29)
- 2.4. Asembler wysokopoziomowy (HLA) (30)
- 2.5. Myśl na wysokim poziomie, pisz na niskim (31)
- 2.6. Paradygmat programowania w asemblerze (myślenie na niskim poziomie) (32)
- 2.7. Asembler. Sztuka programowania i inne materiały (34)

Rozdział 3. Asembler 80x86 dla zwykłego programisty (37)

- 3.1. Dobrze poznać jakiś asembler, ale jeszcze lepiej poznać ich kilka (38)
- 3.2. Składnia asemblera 80x86 (38)
- 3.3. Podstawy architektury 80x86 (39)
    - 3.3.1. Rejestry (39)
    - 3.3.2. Rejestry ogólnego przeznaczenia (40)
    - 3.3.3. Rejestr EFLAGS (41)
- 3.4. Literały stałych (42)
    - 3.4.1. Literały binarne (42)
    - 3.4.2. Literały dziesiętne (42)
    - 3.4.3. Literały szesnastkowe (43)
    - 3.4.4. Literały znakowe i łańcuchowe (44)
    - 3.4.5. Liczbowe literały zmiennoprzecinkowe (45)
- 3.5. Stałe symboliczne w asemblerach (45)
    - 3.5.1. Stałe symboliczne w HLA (46)
    - 3.5.2. Stałe symboliczne w Gas (46)
    - 3.5.3. Stałe symboliczne w MASM i TASM (46)
- 3.6. Tryby adresowania 80x86 (47)
    - 3.6.1. Tryby adresowania rejestrów 80x86 (47)

    - 3.6.2. Bezpośrednie podawanie wartości (48)
    - 3.6.3. Tryb adresowania z przesunięciem (49)
    - 3.6.4. Tryb adresowania pośredniego przez rejestr (50)
    - 3.6.5. Indeksowany tryb adresowania (51)
    - 3.6.6. Skalowane tryby adresowania z indeksowaniem (53)
- 3.7. Deklarowanie danych w językach asemblerowych (55)
    - 3.7.1. Deklarowanie danych w HLA (55)
    - 3.7.2. Deklarowanie danych w MASM i TASM (56)
    - 3.7.3. Deklarowanie danych w Gas (57)
    - 3.7.4. Dostęp do zmiennych bajtowych (57)
- 3.8. Określanie wielkości operandów w asemblerze (59)
    - 3.8.1. Wymuszanie typów w HLA (60)
    - 3.8.2. Wymuszenie typu w MASM i TASM (60)
    - 3.8.3. Wymuszanie typu w Gas (61)
- 3.9. Minimalny zestaw instrukcji 80x86 (61)
- 3.10. Dodatkowe informacje (61)

Rozdział 4. Asembler PowerPC dla zwykłego programisty (63)

- 4.1. Dobrze poznać jakiś asembler, ale jeszcze lepiej poznać ich kilka (64)
- 4.2. Składnia asemblera (64)
- 4.3. Podstawy architektury PowerPC (64)
    - 4.3.1. Rejestry ogólnego przeznaczenia (65)
    - 4.3.2. Zmiennoprzecinkowe rejestry ogólnego przeznaczenia (65)
    - 4.3.3. Rejestry ogólnego przeznaczenia dostępne w trybie użytkownika (65)
- 4.4. Literały stałych (68)
    - 4.4.1. Literały binarne (68)
    - 4.4.2. Literały dziesiętne (69)
    - 4.4.3. Literały szesnastkowe (69)
    - 4.4.4. Literały znakowe i łańcuchowe (69)
    - 4.4.5. Literały zmiennoprzecinkowe (69)
- 4.5. Stałe symboliczne w asemblerze (70)
- 4.6. Tryby adresowania PowerPC (70)
    - 4.6.1. Tryby adresowania rejestrów PowerPC (70)
    - 4.6.2. Bezpośrednie podawanie wartości (70)
    - 4.6.3. Tryby adresowania pamięci PowerPC (71)
- 4.7. Deklarowanie danych w językach asemblerowych (72)
- 4.8. Określanie wielkości operandów w asemblerze (74)
- 4.9. Minimalny zestaw instrukcji (75)
- 4.10. Dodatkowe informacje (75)

Rozdział 5. Narzędzia do analizy wyników kompilacji (77)

- 5.1. Typy plików używanych w językach programowania (78)
- 5.2. Pliki z kodem źródłowym (78)
    - 5.2.1. Pliki źródłowe podzielone na elementy (78)
    - 5.2.2. Specjalne formaty kodu źródłowego (79)
- 5.3. Rodzaje procesorów języków komputerowych (80)
    - 5.3.1. Czyste interpretery (80)
    - 5.3.2. Interpretery (80)

    - 5.3.3. Kompilatory (80)
    - 5.3.4. Kompilatory przyrostowe (81)
- 5.4. Proces translacji (82)
    - 5.4.1. Analiza leksykalna i tokeny (84)
    - 5.4.2. Parsowanie (analiza składniowa) (85)
    - 5.4.3. Generowanie kodu pośredniego (86)
    - 5.4.4. Optymalizacja (87)
    - 5.4.5. Porównanie optymalizacji różnych kompilatorów (97)
    - 5.4.6. Generowanie kodu natywnego (97)
- 5.5. Wyniki kompilacji (97)
    - 5.5.1. Generowanie przez kompilator kodu źródłowego (98)
    - 5.5.2. Generowanie przez kompilator kodu asemblera (99)
    - 5.5.3. Generowanie przez kompilator kodu pośredniego (100)
    - 5.5.4. Generowanie przez kompilator plików wykonywalnych (101)
- 5.6. Formaty plików z kodem pośrednim (101)
    - 5.6.1. Nagłówek pliku COFF (102)
    - 5.6.2. Nagłówek opcjonalny COFF (104)
    - 5.6.3. Nagłówki sekcji COFF (107)
    - 5.6.4. Sekcje COFF (109)
    - 5.6.5. Sekcja relokacji (109)
    - 5.6.6. Informacje o symbolach i dla programu uruchomieniowego (110)
    - 5.6.7. Więcej o formatach plików z kodem pośrednim (110)
- 5.7. Formaty plików wykonywalnych (110)
    - 5.7.1. Strony, segmenty i wielkość pliku (111)
    - 5.7.2. Fragmentacja wewnętrzna (113)
    - 5.7.3. Po co zatem w ogóle oszczędzać na miejscu? (113)
- 5.8. Wyrównanie danych i kodu w pliku z kodem pośrednim (115)
    - 5.8.1. Dobór wielkości wyrównania sekcji (116)
    - 5.8.2. Łączenie sekcji (117)
    - 5.8.3. Sterowanie wyrównaniem sekcji (117)
    - 5.8.4. Wyrównanie sekcji a moduły biblioteczne (118)
- 5.9. Konsolidatory i ich wpływ na kod (125)
- 5.10. Dodatkowe informacje (128)

Rozdział 6. Narzędzia do analizy wyników kompilacji (129)

- 6.1. Tytułem wstępu (130)
- 6.2. Nakazywanie kompilatorowi generowania kodu asemblerowego (131)
    - 6.2.1. Asembler z kompilatorów GNU i Borlanda (131)
    - 6.2.2. Kod asemblerowy z Visual C++ (132)
    - 6.2.3. Przykładowy kod asemblerowy (132)
    - 6.2.4. Analiza asemblerowych wyników kompilacji (141)
- 6.3. Analiza wyników kompilacji za pomocą narzędzi do kodu pośredniego (142)
    - 6.3.1. Narzędzie dumpbin.exe Microsoftu (142)
    - 6.3.2. Program FSF/GNU objdump.exe (154)
- 6.4. Użycie deasemblerów do analizy wyników kompilacji (158)
- 6.5. Użycie programu uruchomieniowego do analizy wyników kompilacji (161)
    - 6.5.1. Użycie programu uruchomieniowego z IDE (161)
    - 6.5.2. Użycie samodzielnego programu uruchomieniowego (162)
- 6.6. Porównywanie wyników dwóch kompilacji (164)

    - 6.6.1. Porównywanie wersji za pomocą narzędzia diff (164)
    - 6.6.2. Porównywanie ręczne (173)
- 6.7. Dodatkowe informacje (174)

Rozdział 7. Stałe a języki wysokiego poziomu (175)

- 7.1. Literały stałych a wydajność programu (176)
- 7.2. Literały stałe a stałe deklarowane (178)
- 7.3. Wyrażenia stałe (179)
- 7.4. Stałe deklarowane a obiekty w pamięci tylko do odczytu (181)
- 7.5. Typy wyliczeniowe (182)
- 7.6. Stałe logiczne (184)
- 7.7. Stałe zmiennoprzecinkowe (186)
- 7.8. Stałe łańcuchowe (191)
- 7.9. Stałe typów złożonych (195)
- 7.10. Dodatkowe informacje (196)

Rozdział 8. Zmienne w językach wysokiego poziomu (197)

- 8.1. Organizacja pamięci w trakcie działania programu (197)
    - 8.1.1. Sekcje kodu, stałych i danych tylko do odczytu (198)
    - 8.1.2. Sekcja zmiennych statycznych (200)
    - 8.1.3. Sekcja BSS (201)
    - 8.1.4. Sekcja stosu (202)
    - 8.1.5. Sekcja sterty i dynamiczna alokacja pamięci (203)
- 8.2. Czym jest zmienna? (204)
    - 8.2.1. Atrybuty (204)
    - 8.2.2. Wiązanie (204)
    - 8.2.3. Obiekty statyczne (204)
    - 8.2.4. Obiekty dynamiczne (205)
    - 8.2.5. Zakres (205)
    - 8.2.6. Czas życia (205)
    - 8.2.7. Czym zatem jest zmienna? (206)
- 8.3. Zmienne w pamięci (206)
    - 8.3.1. Wiązanie statyczne i statyczne zmienne (206)
    - 8.3.2. Wiązanie pseudostatyczne i zmienne automatyczne (210)
    - 8.3.3. Wiązanie dynamiczne i zmienne dynamiczne (213)
- 8.4. Typowe elementarne typy danych (217)
    - 8.4.1. Zmienne całkowitoliczbowe (217)
    - 8.4.2. Zmienne zmiennoprzecinkowe - czyli rzeczywiste (220)
    - 8.4.3. Zmienne znakowe (221)
    - 8.4.4. Zmienne logiczne (222)
- 8.5. Adresy zmiennych a języki wysokiego poziomu (222)
    - 8.5.1. Alokacja pamięci na zmienne globalne i statyczne (223)
    - 8.5.2. Użycie zmiennych automatycznych w celu zmniejszenia przesunięć (224)
    - 8.5.3. Alokacja pamięci na zmienne pośrednie (230)
    - 8.5.4. Alokacja pamięci na zmienne dynamiczne i wskaźniki (231)
    - 8.5.5. Użycie rekordów (struktur) do zmniejszenia przesunięć (233)
    - 8.5.6. Zmienne rejestrowe (235)

- 8.6. Wyrównanie zmiennych w pamięci (236)
  - 8.6.1. Rekordy i wyrównanie (241)
- 8.7. Dodatkowe informacje (246)

Rozdział 9. Tablicowe typy danych (249)

- 9.1. Czym jest tablica? (249)
  - 9.1.1. Deklarowanie tablic (250)
  - 9.1.2. Zapis tablic w pamięci (254)
  - 9.1.3. Dostęp do elementów tablicy (257)
  - 9.1.4. Dopełnianie a kodowanie (259)
  - 9.1.5. Tablice wielowymiarowe (262)
  - 9.1.6. Tablice dynamiczne a tablice statyczne (275)
- 9.2. Dodatkowe informacje (284)

Rozdział 10. Łańcuchowe typy danych (285)

- 10.1. Formaty łańcuchów znakowych (286)
  - 10.1.1. Łańcuchy zakończone zerem (286)
  - 10.1.2. Łańcuchy poprzedzone długością (302)
  - 10.1.3. Łańcuchy 7-bitowe (304)
  - 10.1.4. Łańcuchy HLA (305)
  - 10.1.5. Łańcuchy oparte na deskryptorach (308)
- 10.2. Łańcuchy statyczne, pseudodynamiczne i dynamiczne (309)
  - 10.2.1. Łańcuchy statyczne (309)
  - 10.2.2. Łańcuchy pseudodynamiczne (310)
  - 10.2.3. Łańcuchy dynamiczne (310)
- 10.3. Zliczanie odwołań do łańcuchów (311)
- 10.4. Łańcuchy w Delphi i Kyliksie (311)
- 10.5. Korzystanie z łańcuchów w językach wysokiego poziomu (312)
- 10.6. Dane znakowe w łańcuchach (314)
- 10.7. Dodatkowe informacje (315)

Rozdział 11. Wskaźnikowe typy danych (317)

- 11.1. Definiowanie i odkrywanie tajemnic wskaźników (318)
- 11.2. Implementacje wskaźników w językach wysokopoziomowych (319)
- 11.3. Wskaźniki i dynamiczny przydział pamięci (322)
- 11.4. Operacje na wskaźnikach i arytmetyka wskaźników (323)
  - 11.4.1. Dodawanie liczby całkowitej do wskaźnika (324)
  - 11.4.2. Odejmowanie liczby całkowitej od wskaźnika (326)
  - 11.4.3. Odejmowanie wskaźnika od wskaźnika (327)
  - 11.4.4. Porównywanie wskaźników (328)
  - 11.4.5. Stosowanie logicznych operatorów AND i OR dla wskaźników (330)
  - 11.4.6. Pozostałe operatory wskaźnikowe (331)
- 11.5. Prosty przykład alokatora pamięci (333)
- 11.6. Odzyskiwanie pamięci (336)
- 11.7. Przydział pamięci na poziomie systemu operacyjnego (337)
- 11.8. Dodatkowa złożoność pamięciowa menedżera sterty (338)
- 11.9. Typowe problemy związane z wskaźnikami (341)

    - 11.9.1. Używanie wskaźnika niezainicjalizowanego (341)
    - 11.9.2. Używanie wskaźnika zawierającego nieprawidłową wartość (342)
    - 11.9.3. Korzystanie z bloku pamięci już po jej zwolnieniu (343)
    - 11.9.4. Zaniedbywanie zwalniania niepotrzebnej pamięci (344)
    - 11.9.5. Uzyskiwanie dostępu do danych za pośrednictwem danych błędnego typu (345)
- 11.10. Dodatkowe informacje (346)

Rozdział 12. Rekordy, unie i klasowe typy danych (347)

- 12.1. Rekordy (348)
    - 12.1.1. Deklaracje rekordów w różnych językach programowania (349)
    - 12.1.2. Tworzenie egzemplarza rekordu (350)
    - 12.1.3. Inicjalizacja danych rekordu w czasie kompilacji (357)
    - 12.1.4. Pamięciowe reprezentacje rekordów (361)
    - 12.1.5. Podnoszenie efektywności pamięciowej za pomocą rekordów (364)
    - 12.1.6. Dynamiczne typy rekordowe i bazy danych (366)
- 12.2. Unie wyróżnikowe (367)
- 12.3. Deklaracje unii w różnych językach programowania (368)
    - 12.3.1. Deklaracje unii w języku C/C++ (368)
    - 12.3.2. Deklaracje unii w języku Pascal (Delphi, Kylix) (368)
    - 12.3.3. Deklaracje unii w języku HLA (369)
- 12.4. Pamięciowa reprezentacja unii (370)
- 12.5. Pozostałe zastosowania unii (371)
- 12.6. Typy wariantowe (372)
- 12.7. Przestrzenie nazw (377)
- 12.8. Klasy i obiekty (379)
    - 12.8.1. Klasy kontra obiekty (380)
    - 12.8.2. Prosta deklaracja klasy w języku C++ (380)
    - 12.8.3. Tabele metod wirtualnych (381)
    - 12.8.4. Współdzielenie tabel metod wirtualnych (385)
    - 12.8.5. Dziedziczenie w klasach (386)
    - 12.8.6. Polimorfizm w klasach (389)
    - 12.8.7. Klasy, obiekty i problem wydajności (390)
- 12.9. Dodatkowe informacje (391)

Rozdział 13. Wyrażenia arytmetyczne i logiczne (393)

- 13.1. Wyrażenia arytmetyczne w kontekście architektury komputera (394)
    - 13.1.1. Maszyny stosowe (394)
    - 13.1.2. Maszyny akumulatorowe (399)
    - 13.1.3. Maszyny rejestrowe (401)
    - 13.1.4. Typowe formy zapisu wyrażeń arytmetycznych (403)
    - 13.1.5. Architektury trójadresowe (403)
    - 13.1.6. Architektury dwuadresowe (404)
    - 13.1.7. Różnice architekturalne w kontekście Twojego kodu (404)
    - 13.1.8. Obsługa wyrażeń złożonych (405)
- 13.2. Optymalizacja wyrażeń arytmetycznych (406)
    - 13.2.1. Składanie stałych (407)
    - 13.2.2. Propagacja stałych (408)

    - 13.2.3. Eliminacja martwego kodu (409)
    - 13.2.4. Eliminacja powtarzających się podwyrażeń (411)
    - 13.2.5. Redukcja złożoności wyrażeń (415)
    - 13.2.6. Indukcja (420)
    - 13.2.7. Niezmienne wyrażenia w pętlach (422)
    - 13.2.8. Mechanizmy optymalizujące i programiści (425)
- 13.3. Skutki uboczne stosowania wyrażeń arytmetycznych (427)
- 13.4. Skutki uboczne zawierania - punkty sekwencji (432)
- 13.5. Eliminowanie problemów wynikających z występowania skutków ubocznych (436)
- 13.6. Wymuszanie określonego porządku przetwarzania kodu (437)
- 13.7. Skrócone wyznaczanie wartości wyrażeń arytmetycznych (439)
    - 13.7.1. Skrócone wyznaczanie wartości wyrażeń logicznych (440)
    - 13.7.2. Wymuszanie skróconego lub pełnego wyznaczania wartości wyrażeń logicznych (443)
    - 13.7.3. Skrócone wyznaczanie wartości wyrażeń a kwestia wydajności (444)
- 13.8. Względne koszty operacji arytmetycznych (449)
- 13.9. Dodatkowe informacje (450)

Rozdział 14. Struktury sterujące i decyzje programowe (453)

- 14.1. Struktury sterujące są wolniejsze od wyrażeń arytmetycznych! (454)
- 14.2. Wprowadzenie do niskopoziomowych struktur sterujących (454)
- 14.3. Rozkaz goto (458)
- 14.4. Wyrażenia break, continue, next, return i inne ograniczone formy wyrażenia goto (462)
- 14.5. Wyrażenie if (463)
    - 14.5.1. Optymalizacja niektórych konstrukcji if-else (466)
    - 14.5.2. Wymuszanie pełnego wyznaczania wartości wyrażeń logicznych stosowanych w wyrażeniach if (469)
    - 14.5.3. Wymuszanie skróconego wyznaczania wartości wyrażeń logicznych stosowanych w wyrażeniach if (474)
- 14.6. Wyrażenie switch (case) (480)
    - 14.6.1. Semantyka wyrażenia switch (case) (482)
    - 14.6.2. Tabele skoków kontra sekwencje porównań (482)
    - 14.6.3. Pozostałe implementacje konstrukcji switch (case) (490)
    - 14.6.4. Kod maszynowy generowany przez kompilatory na podstawie wyrażeń switch (501)
- 14.7. Dodatkowe informacje (502)

Rozdział 15. Iteracyjne struktury sterujące (505)

- 15.1. Pętla while (505)
    - 15.1.1. Wymuszanie pełnego wyznaczania wartości wyrażeń logicznych w pętli while (509)
    - 15.1.2. Wymuszanie skróconego wyznaczania wartości wyrażeń logicznych w pętli while (518)
- 15.2. Pętla repeat..until (do..until lub do..while) (521)
    - 15.2.1. Wymuszanie pełnego wyznaczania wartości wyrażeń logicznych w pętli repeat..until (524)

    - 15.2.2. Wymuszanie skróconego wyznaczania wartości wyrażeń logicznych w pętli repeat..until (527)
- 15.3. Pętla forever..endfor (532)
    - 15.3.1. Wymuszanie pełnego przetwarzania wyrażenia logicznego wykorzystywanego w pętli forever (535)
    - 15.3.2. Wymuszanie skróconego przetwarzania wyrażenia logicznego wykorzystywanego w pętli forever (535)
- 15.4. Pętla określona (pętle for) (535)
- 15.5. Dodatkowe informacje (537)

Rozdział 16. Funkcje i procedury (539)

- 16.1. Proste wywołania funkcji i procedur (540)
    - 16.1.1. Składowanie adresu zwracanej wartości (543)
    - 16.1.2. Pozostałe źródła opóźnień (547)
- 16.2. Funkcje-liście i procedury-liście (548)
- 16.3. Makra i funkcje wbudowane (553)
- 16.4. Przekazywanie parametrów do funkcji lub procedury (559)
- 16.5. Rekordy aktywacji i stos (566)
    - 16.5.1. Struktura rekordu aktywacji (569)
    - 16.5.2. Przypisywanie przesunięć zmiennym lokalnym (572)
    - 16.5.3. Wiązanie przesunięć z parametrami (575)
    - 16.5.4. Uzyskiwanie dostępu do parametrów i zmiennych lokalnych (580)
- 16.6. Mechanizmy przekazywania parametrów (589)
    - 16.6.1. Przekazywanie przez wartość (590)
    - 16.6.2. Przekazywanie przez referencję (590)
- 16.7. Wartości zwracane przez funkcje (592)
- 16.8. Dodatkowe informacje (599)

Dodatek A Inżynieria oprogramowania (601)
Dodatek B Krótkie zestawienie informacji o rodzinach procesorów 80x86 oraz PowerPC (603)

- B.1. Różnice architekturalne pomiędzy technologiami RISC i CISC (604)
    - B.1.1. Zakres zadań realizowanych przez pojedyncze rozkazy (604)
    - B.1.2. Rozmiar rozkazu (605)
    - B.1.3. Częstotliwość taktowania zegara i współczynnik liczby cykli na rozkaz (606)
    - B.1.4. Dostęp do pamięci i tryby adresowania (607)
    - B.1.5. Rejestry (608)
    - B.1.6. Operandy bezpośrednie (stałe) (608)
    - B.1.7. Stosy (609)
- B.2. Kompilatory i interfejs ABI (609)
- B.3. Profesjonalne programowanie dla obu architektur (610)

Dodatek C Dodatki internetowe (613)
Skorowidz (615)