Dokumentacja Projektu

Architektura Systemów Komputerowych

NWW w NASM

Mateusz Bocak 125104
Informatyka II rok
Uniwersytet Rzeszowski

Spis treści

Opis

Projekt w asemblerze NASM oblicza Najmniejszą Wspólną Wielokrotność (NWW) dwóch liczb całkowitych w systemie Windows przy użyciu środowiska uruchomieniowego `asmloader`. Program korzysta z 32-bitowej architektury Intel i wykorzystuje API `asmloader` do operacji wejścia/wyjścia.

Zależności

- `NASM` - Netwide Assembler, do kompilacji kodu źródłowego.
- `asmloader` - środowisko uruchomieniowe wymagane do wykonania programu.
- `GCC` – GNU Compiler Collection – kompilator GNU służący do kompilowania pliku obiektowego na formę wykonywalną.

Struktura Pliku

Plik składa się z kilku sekcji:

1. **Sekcja pobierania danych:** Pierwsze dwie sekcje pliku nww.asm pobierają dane i wyświetlają odpowiednie komunikaty na STD.
2. **Sekcja wykonująca obliczenia:** Kolejne sekcje odpowiadają za przeprowadzenie obliczeń

Instrukcja Kompilacji

Aby skompilować program, wykonaj poniższe kroki:

1. Zapisz kod źródłowy w pliku z rozszerzeniem `.asm`, na przykład `nww.asm`.
2. Skorzystaj z NASM do skompilowania kodu do formatu binarnego:

```
nasm -o nww.asm nww.o -f  win32
```

Instrukcja Kompilacji do pliku wykonywalnego

1. Zapisz kod źródłowy w pliku z rozszerzeniem `.asm`, na przykład `nww.asm`.
2. Skorzystaj z kompilatora gcc do skompilowania kodu do formatu wykonywalnego:

```
nasm -o nww_do_exe.asm nww_do_exe.o -f  win32
gcc -o nww_do_exe.o nww_do_exe.exe -m32
```

Instrukcja Uruchomienia

1. Wejdź do folderu zawierającego projekt `nww`.
2. Uruchom skompilowany plik wykonywalny za pomocą:

```
.\nww.exe
```

API `asmloader`

Program wykorzystuje następujące funkcje API:

- `0` - exit
- `1` - putchar
- `2` - getchar
- `3` - printf
- `4` – scanf

Opis Kodu

```
          [bits 32]

; Inicjalizacja stosu i wywołanie pobierania pierwszej liczby
          call geta
fora:
          db "a = ", 0
geta:
;         esp -> [fora][ret]
          call [ebx+3*4]

;         esp -> [a][ret]

          push esp

;         esp -> [addr_a][a][ret]

          call geta2
fora2:
          db "%i", 0
geta2:
;         esp -> [fora2][addr_a][a][ret]
          call [ebx+4*4]

          add esp, 2*4
;         esp -> [a][ret]

          pop esi;
          mov eax, esi
          ;      esi = a
;         esp -> [ret]

; Pobieranie drugiej liczby
          call getb
forb:
          db "b = ", 0
getb:
;         esp -> [forb][ret]
          call [ebx+3*4]

;         esp -> [b][ret]

          push esp

;         esp -> [addr_b][b][ret]

          call getb2
forb2:
          db "%i", 0
getb2:
;         esp -> [forb2][addr_b][b][ret]
          call [ebx+4*4]

          add esp, 2*4

;         esp -> [b][ret]

          pop edi
          push edi
```

```
        push esi

        push esi;     esp -> [esi][esi][edi][ret]
        push edi;     esp -> [edi][esi][ret]

        call wypisz

; Wyświetlanie formatu NWW
format:
        db "NWW(%i,%i) = ", 0
wypisz:
;       esp -> [format][edi][esi][ret]
        call [ebx+3*4]

        add esp, 3*4
;       esp -> [ret]

; Algorytm obliczania NWW
start:
        cmp esi, edi
        je koniec
        ja opcja1

        sub edi, esi
        jmp start


opcja1:
        sub esi, edi
        jmp start

; Wyjście i wyświetlenie wyniku
koniec:
        pop eax

        div esi

        pop ecx

        mul ecx
        push eax;     esp -> [esi][ret]
        call wypisz2
format2:
        db "%i", 0xA, 0
wypisz2:
;       esp -> [format2][esi][ret]
        call [ebx+3*4]

        add esp, 2*4
;       esp -> [ret]

        push 0

        call [ebx+0*4]
```

Szczegółowy Opis

1. **Pobieranie danych**:
   - `call geta`: Pobiera pierwszą liczbę i zapisuje ją w `esi`.
   - `call getb`: Pobiera drugą liczbę i zapisuje ją w `edi`.
2. **Wyświetlanie danych**:
   - `call wypisz`: Wyświetla format NWW przed obliczeniami.
3. **Obliczanie NWW**:
   - Algorytm wykorzystuje podejście podobne do algorytmu Euklidesa, iteracyjnie obliczając NWD i wykorzystując wynik do wyznaczenia NWW.
4. **Wyjście**:
   - `call wypisz2`: Wyświetla wynik końcowy.
   - `call [ebx+0*4]`: Zamyka program.

Podsumowanie

Ten program ilustruje podstawowe techniki programowania w asemblerze na platformie Windows przy użyciu `asmloader`.

W projekcie wykonywane są podstawowe operacje na rejestrach - obliczając NWW, oraz podstawowe operacje na stosie (pop, push). Dodatkowo jest wykonywana „sztuczka" z wywołaniem `scanf` i `call` do pobierania danych od użytkownika. Polega ona na sposobie działania „`call`", które zawsze po powrocie wskazuje na kolejną linie, które z kolei wywołują wyświetlenie danych na standardowym wyjściu.

Kod jest zoptymalizowany pod kątem prostoty i czytelności, a wszystkie operacje są wykonywane w kontekście 32-bitowej architektury Intel.