

# Ćwiczenie JCC-A01. Asembler, arytmetyka 32-bitowa

## JĘZYK C I C++ W PROGRAMACH STEROWANIA LABORATORIUM SYSTEMÓW STEROWNIA PRZEMYSŁOWEGO I AUTOMATYKI BUDYNKÓW

KATEDRA AUTOMATYKI NAPĘDU I URZĄDZEŃ PRZEMYSŁOWYCH  
WWW.KANUP.AGH.EDU.PL

AKADEMIA GÓRNICZO-HUTNICZA  
WWW.AGH.EDU.PL

**Temat:** Asembler, arytmetyka 32-bitowa

**Narzędzia:** Kompilator, linker, debugger

### Wstęp

Celem ćwiczenia jest napisanie zestawu funkcji wykonujących podstawowe operacje arytmetyczne na 32-bitowych liczbach całkowitych za znakiem w sposób nie powodujący wystąpienia przepełnień i innych efektów ubocznych związanych z przekroczeniem zakresu wyniku dodawania, odejmowania albo mnożenia dwóch liczb 32-bitowych w wynikowym słowie 32-bitowym.

Należy pamiętać, że procesor w czasie operacji dodawania, odejmowania, mnożenia i dzielenia liczb całkowitych nie ogranicza wyniku do maksymalnej pojemności danego słowa. Fakt wystąpienia przepełnienia można stwierdzić, testując odpowiednie flagi ustawiane zależnie od wyniku operacji wykonywanej przez procesor. Kod maszynowy generowany przez kompilatory nie uwzględnia tego faktu.

Opisane zjawisko jest niebezpieczne w mikrokomputerowych układach regulacji, gdyż może spowodować nieoczekiwane zachowanie układu. Przepełnienie może np. wystąpić w regulatorze PID przy długotrwałym całkowaniu błędu różnego od zera.

### Sprawdzanie stanu przepełnienia

W celu sprawdzenia, czy po operacji dodawania, odejmowania, mnożenia nie wystąpiło przepełnienie należy sprawdzać podstawowe flagi procesora. W przypadku procesorów Intel'a z rodziny zgodnych z architekturą x86 (8086, ..., 80386, 80486, Pentium, ..., Pentium4) konieczne jest testowanie następujących flag:

- C (Carry) – flaga przeniesienia jest ustawiana, gdy przy operacji dodawania lub odejmowania jest generowane przeniesienie (lub pożyczka) z najstarszego bitu. Flaga ta jest wykorzystywana w arytmetyce wielokrotnej precyzji. Ustawienie tej flagi oznacza także przepełnienie w arytmetyce bez znaku.
- O (Overflow) – flaga przepełnienia jest ustawiana, gdy nastąpi przepełnienie przy operacji dodawania i odejmowania w arytmetyce ze znakiem. Flaga O jest również ustawiana poprawnie w przypadku mnożenia ze znakiem jak i bez znaku, gdy poprawny wynik jest dłuższy od argumentów (np. mnożymy zawartość dwóch rejestrów 32-bitowych, a wynik się nie mieści w rejestrze 32-bitowym)
- S (Sign) – flaga znaku jest ustawiana zgodnie z najstarszym bitem wyniku, który oznacza znak w arytmetyce ze znakiem. W przypadku obliczeń w arytmetyce bez znaku flaga ta nie ma praktycznego znaczenia.

W celu sprawdzenia przepełnienia należy przetestować po danej operacji flagę O. Do tego celu służą instrukcje asemblerowe skoków warunkowych:

```
JO lab ; skok do etykiety lab;, gdy flaga O=1
JNO lab ; skok do etykiety lab;, gdy flaga O=0
```

Do sprawdzania stanu pozostałych flag można następujących instrukcji skoków warunkowych:

```
JC lab ; skok do etykiety lab;, gdy flaga C=1
JNC lab ; skok do etykiety lab;, gdy flaga C=0
JS lab ; skok do etykiety lab;, gdy flaga S=1
JNS lab ; skok do etykiety lab;, gdy flaga S=0
```

### Funkcje w języku asemblera w programie w C (C++)

W programie pisanym w C lub C++ w środowisku systemu QNX 4.2x można pojedyncze funkcje zakodować stosując język asemblera procesora. Sposób pisania takiej funkcji zostanie przedstawiony na przykładzie funkcji realizującej mnożenie dwóch argumentów. W przypadku przepełnienia wynik będzie ustawiany na maksymalną albo minimalną liczbę, jaką można przedstawić w typie int, zależnie od kierunku, w który nastąpiło przepełnienie.

```
long int Mul(long int, long int); // prototyp funkcji
#pragma aux Mul = \ // nagłówek funkcji w asemblerze
    "imul edx" \ // mnożenie edx:eax = eax * edx
```

## Ćwiczenie JCC-A01. Asembler, arytmetyka 32-bitowa

```
"jno l1"           \           // jeśli brak przepełnienia – koniec
"or edx, edx"     \           // ustawieniu flag
"jns l2"          \           // gdy wynik >0, skocz do l2
"mov eax,80000000h" \       // wstaw do eax minimalną liczbę
"jmp l1"          \           // skocz do l1
"|2:  mov eax, 7FFFFFFFh" \   // wstaw do eax maksymalną liczbę
"|1:"             \
parm [eax] [edx]  \           // wskazanie, w których rejestrach są argumenty
value [eax]       \           // wskazanie, w którym rejestrze jest wynik
```

Uwagi:

1. W procesorach rodziny x86 są do dyspozycji 4 podstawowe rejestry 32-bitowe: EAX, EBX, ECX i EDX. Te rejestry jest najwygodniej wykorzystywać do obliczeń.
2. Instrukcja `IMUL rejestr` wykonuje operację  $EDX:EAX = EAX * \textit{rejestr}$ . Oznacza to, że jeden z argumentów musi być w rejestrze EAX, drugi jest w innym rejestrze. Wynik jest umieszczany w dwóch rejestrach: starsza część w rejestrze EDX, młodsza – w EAX. Flaga Overflow (O) jest ustawiana, gdy wynik nie jest w stanie zmieścić się tylko w rejestrze EAX. Należy pamiętać, że instrukcja `IMUL` ustawia poprawnie tylko flagi O i C (zawsze tak samo). Pozostałe flagi są ustawiane w sposób nieokreślony.
3. Instrukcja `OR EDX, EDX` jest wykonana w celu tylko ustawienia flagi znaku S. Ponieważ najstarszy bit w starszej części wyniku niesie informację o znaku wyniku, to w celu ustawienia flag wykonana jest instrukcja `OR EDX, EDX`, gdyż nie zmienia ona zawartości rejestru EDX. Dalej testując flagę S można określić poprawny znak wyniku.

W funkcji obliczających sumę i różnicę z wykrywaniem przepełnień należy wykorzystać instrukcje asemblerowe:

```
ADD EAX, rejestr
SUB EAX, rejestr
```

w celu wykonania dodawania i odejmowania.

### Program ćwiczenia

Napisać w asemblerze funkcje:

```
long int Plus( long int x1, long int x2 );
long int Minus( long int x1, long int x2 );
long int Mul( long int x1, long int x2 );
```

które obliczać będą sumę, różnicę i iloczyn argumentów z ograniczeniem wyniku. Jeżeli wynik nie mieści się w słowie 32-bitowym to funkcje te mają zwracać liczbę `-2147483648 (80000000h)` albo `+2147483647 (7FFFFFFFh)`, zależnie czy przekroczenie zakresu nastąpiło w kierunku liczb ujemnych czy dodatnich.

Napisać w języku C funkcję

```
long int Div( long int x1, long int x2 );
```

która oblicza iloraz całkowity argumentów. Jeżeli dzielnik (`x2`) jest równy zero, to wynik dzielenia ma być równy:

- `+2147483647 (7FFFFFFFh)`, gdy dzielna `x1` jest dodatnia,
- `0`, gdy dzielna `x1` jest równa 0,
- `-2147483648 (80000000h)`, gdy dzielna `x1` jest ujemna.

W pozostałych przypadkach funkcja zwraca normalny wynik dzielenia.

Napisać w języku C funkcje

```
void Inc(long int *x);
void Dec(long int *x);
```

realizującą inkrementację i dekrementację zmiennej zgodnie z wcześniej opisanymi zasadami dodawania i odejmowania. Najwygodniej użyć w tym celu wcześniej napisanych funkcji `Plus( )` i `Minus( )`.

Napisać według własnego projektu program, który pozwoli zaprezentować i sprawdzić działanie wyżej wymienionych funkcji.