Zadanie A: Sortowanie

Zaimplementuj funkcję o sygnaturze

```
void sort(long * data, unsigned long count);
```

która posortuje tablicę `count` liczb pod adresem `data`.

Sortowanie musi działać w czasie $O(n \log n)$ – sugerowane algorytmy to *merge-sort* lub *quick-sort* (w przypadku tego drugiego jako element dzielący należy wybrać *środkowy* element tablicy). Rozwiązanie powinno być w całości zaimplementowane w 64-bitowym assemblerze (nasm). W szczególności nie wolno korzystać z żadnych funkcji systemowych i bibliotecznych – pamięć roboczą można alokować na stosie.

Szablon rozwiązania:

```
BITS        64
SECTION     .text

GLOBAL      sort
sort:
    ; adres tablicy w rdi, długość w rsi
    ret
```

W wypadku korzystania z rejestrów `rbx`, `rbp`, oraz `r12`–`r15`, należy zapisać ich wartość na stosie i przywrócone przed powrotem z funkcji!

Do testowania swojego rozwiązania możesz wykorzystać następujący kod w C++:

```
#include <cstdio>
using namespace std;

extern "C" void sort(long * data, unsigned long count);

int main()
{
  unsigned long count;
  scanf("%lu", &count);
  long data[count];
  for (int i=0; i<count; ++i)
    scanf("%ld", &data[i]);
  sort(data, count);
  for (int i=0; i<count; ++i)
    printf("%ld ", data[i]);
  printf("\n");
  return 0;
}
```

Do kompilacji i linkowania wykorzystaj polecenia:

```
nasm sort.asm -felf64 -o sort.o
g++ -c test.cpp -o test.o
g++ test.o sort.o -o test
```