# Zadania

### Zadanie 0: Porównanie różnych poziomów optymalizacji

### Zadanie 1: Zmienne globalne oraz lokalne

rejestry xmm - 128 bitowe rejestry, używane głównie (ale nie tylko) dla liczb zmiennoprzecinkowych

movss - move scalar single-precision - przenoszenie float miedzy xmm lub 32 bitowej pamięci

addss - add scalar single-precision - dodawanie float

lea - load effective addres - wykonuje działanie wstawiając wynik pod wskazany adres, wykonuje się na jednostce adresowej AGU, a nie na ALU (arytmetyczno-logiczna)

### Zadanie 2: Pętle

movdqa - move aligned double quadword - przenosi wyrównaną (aligned) wartość double

pxor - xor dla xmm

shr - shift right - mnoży przez 2 podaną iliść razy

paddd - add packed doubleword integers - dodaje integery spakowane w podanych adresach (2 na adres (doubleword))

PSRLDQ - Shift Double Quadword Right Logical - przesunięcie bitowe na xmm podaną ilość razy

jle - jump short if less or equal

imul - signed multiply, mul - unsigned multiply

### Zadanie 3: Instrukcje warunkowe

cmovl - conditional move if less

nop - no operation - nic nie robi, ale przydaje się do wielu rzeczy

### Zadanie 4?: Eliminacja wspólnych podwyrażeń

cdq - convert doubleword to quadword

shr - shift right, zasze wstawia 0 do najstarszego bitu, SAR pozostawia bit znaku

### Zadanie 5: Optymalizacje O0

Przedstawiony problem jest niezdefiniowany przez standard, tj. kompilator może wykonać optymalizacje, ale nie musi, nawet na O0, jednak oba kompilatory (Clang oraz gcc) wykonują optymalizacje. Używanie const\_cast zawsze jest złe, a już na pewno w celu zmiany wartości danej stałej (raczej do przekazywania do funkcji przyjmujacych non-const, ale nie zmieniajacych wartosci)

### Zadanie 6: Ofast

movsd - move scalar double precision

addsd - add scalar double precision

### Zadanie 7: Proste optymalizacje a optymalizacja kompilatora

cdqe - convert doubleword to quadword

### Zadanie 8: Link Time Optimization

### 