Politechnika Częstochowska
Katedra Inteligentnych Systemów Informatycznych

Programowanie Niskopoziomowe

Laboratorium 4

Operacje na skalarach i wektorach

dr inż. Bartosz Kowalczyk

Częstochowa, 14 marca 2023

Spis treści

1 Założenia

Dla zwiększenia czytelności przedłożonej listy zadań, zostały zastosowane aliasy dla typów zmiennych. W języku C++ alias dla dowolnego typu można utworzyć przy użyciu słowa kluczowego `typedef`, które posiada następującą składnię:

Listing 1: Składnia polecenia `typedef` w języku C++

```
typedef <typ-wbudowany> <nowa-nazwa-typu-(alias)>
```

Aliasy typów pokazane w listingu 2 mają zastosowanie do wszystkich zadań niniejszego laboratorium.

Listing 2: Aliasy dla typów zmiennych użytych w zadaniach

```
typedef unsigned char uchar;
typedef unsigned short ushort;
typedef unsigned int uint;
typedef long long int64;
typedef unsigned long long uint64;
```

2 Instrukcje `MOVSX` oraz `MOVSXD`

Korzystając z instrukcji `MOVSX` oraz `MOVSXD` dokonaj niezbędnych konwersji i oblicz wartość wyrażenia $y = \frac{7\,(a+b)}{5}$ przy następujących założeniach:

1. Argumenty `a` i `b` typu `char`, wartość zwracana `y` typu `char`.
2. Argumenty `a` i `b` typu `char`, wartość zwracana `y` typu `short`.
3. Argumenty `a` i `b` typu `char`, wartość zwracana `y` typu `int`.
4. Argumenty `a` i `b` typu `char`, wartość zwracana `y` typu `int64`.
5. Argumenty `a` i `b` typu `uchar`, wartość zwracana `y` typu `char`.
6. Argumenty `a` i `b` typu `uchar`, wartość zwracana `y` typu `uchar`.
7. Argumenty `a` i `b` typu `uchar`, wartość zwracana `y` typu `short`.
8. Argumenty `a` i `b` typu `uchar`, wartość zwracana `y` typu `ushort`.
9. Argumenty `a` i `b` typu `uchar`, wartość zwracana `y` typu `int`.
10. Argumenty `a` i `b` typu `uchar`, wartość zwracana `y` typu `uint`.
11. Argumenty `a` i `b` typu `uchar`, wartość zwracana `y` typu `int64`.
12. Argumenty `a` i `b` typu `uchar`, wartość zwracana `y` typu `uint64`.
13. Argumenty `a` i `b` typu `short`, wartość zwracana `y` typu `short`.
14. Argumenty `a` i `b` typu `short`, wartość zwracana `y` typu `int`.
15. Argumenty `a` i `b` typu `short`, wartość zwracana `y` typu `int64`.
16. Argumenty `a` i `b` typu `ushort`, wartość zwracana `y` typu `short`.
17. Argumenty `a` i `b` typu `ushort`, wartość zwracana `y` typu `ushort`.
18. Argumenty `a` i `b` typu `ushort`, wartość zwracana `y` typu `int`.
19. Argumenty `a` i `b` typu `ushort`, wartość zwracana `y` typu `uint`.
20. Argumenty `a` i `b` typu `ushort`, wartość zwracana `y` typu `int64`.
21. Argumenty `a` i `b` typu `ushort`, wartość zwracana `y` typu `uint64`.
22. Argumenty `a` i `b` typu `int`, wartość zwracana `y` typu `int`.
23. Argumenty `a` i `b` typu `int`, wartość zwracana `y` typu `int64`.
24. Argumenty `a` i `b` typu `uint`, wartość zwracana `y` typu `int`.
25. Argumenty `a` i `b` typu `uint`, wartość zwracana `y` typu `uint`.
26. Argumenty `a` i `b` typu `uint`, wartość zwracana `y` typu `int64`.

27. Argumenty `a` i `b` typu `uint`, wartość zwracana `y` typu `uint64`.
28. Argumenty `a` i `b` typu `int64`, wartość zwracana `y` typu `int64`.
29. Argumenty `a` i `b` typu `uint64`, wartość zwracana `y` typu `int64`.
30. Argumenty `a` i `b` typu `uint64`, wartość zwracana `y` typu `uint64`.

3 Przekazywanie argumentów do funkcji w trybie x64

Przekaż do procedury w języku asembler podane zmienne. Jeżeli to konieczne, dokonaj ich konwersji. Następnie oblicz wartość podanych wyrażeń:

1. $y = a + b + c + d + e + f$
2. $y = a - b - c - d - e - f$
3. $y = ab - cd - e + fg + h$
4. $y = a - bc - d + ef - gh$
5. $y = 128a + 64b + 32c + 16d + 8e + 4f$
6. $y = \frac{a}{2} - \frac{b}{4} + \frac{c}{8} - \frac{d}{16} + \frac{e}{32} - \frac{f}{64}$
7. $y = \frac{10a - b + 40c - d + 11e - f}{10 + 2g - 4h}$, gdzie $10 + 2g - 4h \neq 0$
8. $y = 46ab + 128c \% (12d + 7e + f)$, gdzie $12d + 7e + f \neq 0$

Implementację powyższych funkcji należy rozważyć w następujących scenariuszach:

1. Argumenty `a`, `b`, `c`, `d`, `e`, `f`, `g`, `h` typu `short`, wartość zwracana `y` typu `short`.
2. Argumenty `a`, `b`, `c`, `d`, `e`, `f`, `g`, `h` typu `short`, wartość zwracana `y` typu `int`.
3. Argumenty `a`, `b`, `c`, `d`, `e`, `f`, `g`, `h` typu `short`, wartość zwracana `y` typu `int64`.
4. Argumenty `a`, `b`, `c`, `d`, `e`, `f`, `g`, `h` typu `int`, wartość zwracana `y` typu `int`.
5. Argumenty `a`, `b`, `c`, `d`, `e`, `f`, `g`, `h` typu `int`, wartość zwracana `y` typu `int64`.
6. Argumenty `a`, `b`, `c`, `d`, `e`, `f`, `g`, `h` typu `int64`, wartość zwracana `y` typu `int64`.

4 Operacje skalarne na wektorach

Przekaż do procedury w języku asembler podane wektory. Jeżeli to konieczne, dokonaj konwersji ich elementów. Następnie oblicz wartość podanych wyrażeń:

1. (Suma elementów wektora) $y = \text{sum}(\mathbf{a}) = \sum_{i=0}^{n} a_i$, gdzie $\mathbf{a} \in \mathbb{Z}^n$
2. (Iloczyn elementów wektora) $y = \text{prod}(\mathbf{a}) = \prod_{i=0}^{n} a_i$, gdzie $\mathbf{a} \in \mathbb{Z}^n$
3. (Wartość minimalna wektora) $y = \min(\mathbf{a})$, gdzie $\mathbf{a} \in \mathbb{Z}^n$
4. (Wartość maksymalna wektora) $y = \max(\mathbf{a})$, gdzie $\mathbf{a} \in \mathbb{Z}^n$
5. (Wartość średnia wektora) $y = \text{avg}(\mathbf{a}) = \frac{\sum_{i=0}^{n} a_i}{n}$, gdzie $\mathbf{a} \in \mathbb{Z}^n$
6. (Iloczyn skalarny wektorów) $y = \mathbf{a} \cdot \mathbf{b} = \sum_{i=0}^{n} a_i b_i$, gdzie $\mathbf{a}, \mathbf{b} \in \mathbb{Z}^n$
7. $y = \sum_{i=0}^{n} 5a_i - 3$, gdzie $\mathbf{a} \in \mathbb{Z}^n$
8. $y = \sum_{i=0}^{n} 16a_i + 6$, gdzie $\mathbf{a} \in \mathbb{Z}^n$
9. $y = \sum_{i=0}^{n} \frac{a_i + 6}{4}$, gdzie $\mathbf{a} \in \mathbb{Z}^n$
10. Policz ile elementów parzystych znajduje się w wektorze $\mathbf{a} \in \mathbb{Z}^n$.
11. Policz ile elementów nieparzystych znajduje się w wektorze $\mathbf{a} \in \mathbb{Z}^n$.
12. Policz ile elementów podzielnych przez 8 znajduje się w wektorze $\mathbf{a} \in \mathbb{Z}^n$.
13. Policz ile elementów większych od 0 znajduje się w wektorze $\mathbf{a} \in \mathbb{Z}^n$.
14. Policz ile elementów z przedziału $a_i \in (5, 15]$ znajduje się w wektorze $\mathbf{a} \in \mathbb{Z}^n$.

Implementację powyższych funkcji należy rozważyć w następujących scenariuszach:

1. Wektory `a` i `b` przechowują liczby typu `short`, wartość zwracana `y` typu `short`.
2. Wektory `a` i `b` przechowują liczby typu `short`, wartość zwracana `y` typu `int`.
3. Wektory `a` i `b` przechowują liczby typu `short`, wartość zwracana `y` typu `int64`.
4. Wektory `a` i `b` przechowują liczby typu `int`, wartość zwracana `y` typu `int`.
5. Wektory `a` i `b` przechowują liczby typu `int`, wartość zwracana `y` typu `int64`.
6. Wektory `a` i `b` przechowują liczby typu `int64`, wartość zwracana `y` typu `int64`.

5 Operacje na wektorach

Przekaż do procedury w języku asembler podane wektory. Jeżeli to konieczne, dokonaj konwersji ich elementów. Następnie oblicz wartość podanych wyrażeń:

1. $\mathbf{y} = 16\mathbf{a} + 5 \Rightarrow y_i = 16a_i + 5$, gdzie $i \in [0, \ldots n]$ oraz $\mathbf{a}, \mathbf{y} \in \mathbb{Z}^n$
2. $\mathbf{y} = \mathbf{a}^2 \Rightarrow y_i = a_i^2$, gdzie $i \in [0, \ldots n]$ oraz $\mathbf{a}, \mathbf{y} \in \mathbb{Z}^n$
3. $\mathbf{y} = \mathbf{a} + \mathbf{b} \Rightarrow y_i = a_i + b_i$, gdzie $i \in [0, \ldots n]$ oraz $\mathbf{a}, \mathbf{b}, \mathbf{y} \in \mathbb{Z}^n$
4. $\mathbf{y} = \dfrac{5\mathbf{a} + 4\mathbf{b}}{3} \Rightarrow y_i = \dfrac{5a_i + 4b_i}{3}$, gdzie $i \in [0, \ldots n]$ oraz $\mathbf{a}, \mathbf{b}, \mathbf{y} \in \mathbb{Z}^n$
5. $\mathbf{y} = \dfrac{\mathbf{a}}{\mathbf{b}} \Rightarrow y_i = \dfrac{a_i}{b_i}$, gdzie $b_i \neq 0$, $i \in [0, \ldots n]$ oraz $\mathbf{a}, \mathbf{b}, \mathbf{y} \in \mathbb{Z}^n$
6. $\mathbf{y} = \mathbf{a}\%\mathbf{b} \Rightarrow y_i = a_i\%b_i$, gdzie $b_i \neq 0$, $i \in [0, \ldots n]$ oraz $\mathbf{a}, \mathbf{b}, \mathbf{y} \in \mathbb{Z}^n$
7. Wyzeruj *in situ* elementy wektora $\mathbf{a} \in \mathbb{Z}^n$ o parzystych indeksach.
8. Wyzeruj *in situ* elementy wektora $\mathbf{a} \in \mathbb{Z}^n$ o nieparzystych indeksach.
9. Wyzeruj elementy wektora $\mathbf{a} \in \mathbb{Z}^n$ o parzystych indeksach. Wynik umieścić w wektorze wynikowym $\mathbf{y} \in \mathbb{Z}^n$.
10. Wyzeruj elementy wektora $\mathbf{a} \in \mathbb{Z}^n$ o nieparzystych indeksach. Wynik umieścić w wektorze wynikowym $\mathbf{y} \in \mathbb{Z}^n$.

Implementację powyższych funkcji należy rozważyć w następujących scenariuszach:

1. Wektory `a` i `b` przechowują liczby typu `short`, wektor wynikowy `y` typu `short`.
2. Wektory `a` i `b` przechowują liczby typu `short`, wektor wynikowy `y` typu `int`.
3. Wektory `a` i `b` przechowują liczby typu `short`, wektor wynikowy `y` typu `int64`.
4. Wektory `a` i `b` przechowują liczby typu `int`, wektor wynikowy `y` typu `int`.
5. Wektory `a` i `b` przechowują liczby typu `int`, wektor wynikowy `y` typu `int64`.
6. Wektory `a` i `b` przechowują liczby typu `int64`, wektor wynikowy `y` typu `int64`.