Projekty - Architektura systemów komputerowych

Każdy projekt ma przypisaną ocenę referencyjną. Projekty realizowane są w parach. Jeśli to możliwe, to osoby z pary powinny należeć do tej samej grupy ocenowej. Każda para wykonuje dwa projekty. Każda osoba z pary jest raz programistą a raz testerem. Gotowy program, najpóźniej w dniu deadline'u, przysyła tester z opisem błędów lub informacją, że program działa poprawnie. Projekt ma być dokładnie opisany w komentarzach i przygotowany w dwóch wersjach, jedna dla asmloader'a a druga wykonywalna. Wszystkie programy odczytują dane wejściowe z konsoli. Obrony projektów odbędą się w czasie sesji.

Deadline: 15.06.2026

Projekt 1 (3.0)

Napisz program `podzielniki.asm` wyznaczający wszystkie dodatnie podzielniki liczby całkowitej `n`. Przykładowa sesja:

```
n = -12
podzielniki: 1, 2, 3, 4, 6, 12
```

Projekt 2 (3.0)

Napisz program `nwd.asm` obliczający największy wspólny dzielnik dwóch liczb. Przykładowa sesja:

```
a = 12
b = 8

nwd(8,12) = 4
```

Projekt 3 (3.0)

Napisz program `nww.asm` obliczający najmniejszą wspólną wielokrotność dwóch liczb. Przykładowa sesja:

```
a = 3
b = 4

nww(3,4) = 12
```

Projekt 4 (3.0)

Napisz program `reszta.asm` rozwiązujący problem wydawania reszty z wykorzystaniem możliwie najmniejszej liczby monet. Przykładowa sesja:

```
zl = 13
gr = 28
```

```
Reszta: 13.28
Monety: 5.0 5.0 2.0 1.0 0.2 0.05 0.02 0.01
```

Projekt 5 (3.0)

Napisz program `palindrom.asm` sprawdzający, czy podane zdanie jest palindromem. Przykładowa sesja:

```
zdanie: A to kanapa pana Kota.

zdanie jest palindromem
```

Projekt 6 (3.0)

Napisz program `gra.asm` implementujący grę w kamień, papier i scyzoryk. Przykładowa sesja:

```
1. Kamien
2. Papier
3. Scyzoryk
Wybierz przedmiot: 2

Komputer wylosowal kamien i przegral.
```

Projekt 7 (3.5)

Napisz program `trojki.asm` wyznaczający wszystkie trójki pitagorejskie `(x,y,z)`, gdzie `x < y` oraz liczby `x,y,z` należą do przedziału od 1 do `n`. Przykładowa sesja:

```
n = 15

3 4 5
5 12 13
6 8 10
9 12 15
```

Projekt 8 (3.5)

Napisz program `pierwsze.asm` wyznaczający wszystkie liczby pierwsze z przedziału od 1 do `n`. Przykładowa sesja:

```
n = 10
pierwsze: 2, 3, 5, 7
```

Projekt 9 (3.5)

Napisz program `sito.asm` wyznaczający wszytkie liczby pierwsze z przedziału od 1 do `n` metodą sita Eratostenesa. Przykładowa sesja:

```
n = 10
pierwsze: 2, 3, 5, 7
```

Projekt 10 (3.5)

Napisz program `rozklad.asm` wyznaczający rozkład danej liczby naturalnej `n` na czynniki pierwsze. W implementacji można wykorzystać tablicę liczb pierwszych. Przykładowa sesja:

```
n = 4350
rozklad: 2 3 5 5 29
```

Projekt 11 (3.5)

Napisz program `rownanie.asm` obliczający rozwiązania równania o współczynnikach rzeczywistych `a*x^2 + b*x + c = 0` przy pomocy FPU. Przykładowa sesja:

```
a = 0
b = 0
c = 1

x nalezy do zbioru pustego
```

Projekt 12 (3.5)

Napisz program `square-root.asm` obliczający pierwiastki metodą Newtona-Raphsona z zadaną dokładnością. Przykładowa sesja:

```
x = 5.0
e = 0.01

sqrt(5.0) = 2.238095
```

Projekt 13 (3.5)

Napisz program `cube-root.asm` obliczający pierwiastek trzeciego stopnia z $x$metodą Newtona-Raphsona z zadaną dokładnościąe. Jeżeli y jest przybliżeniem$\sqrt[3]{x}$, to kolejnelepsze przybliżenie uzyskujemy ze wzoru$1/3 \cdot (2y + x/y^2)$. Wynik obliczeń jest satysfakcjonujący jeśli y^3różni się od x nie więcej niż o zadaną dokładność e.

```
x = 5.0
e = 0.01

cbrt(5.0) = 1.709976
```

Projekt 14 (3.5)

Napis progam `bubble.asm` implementujący sortowanie bąbelkowe. Przykładowa sesja:

```
liczby: 3 2 7 5 8
posortowane: 2 3 5 7 8
```

Projekt 15 (3.5)

Napisz program `rozwiniecie.asm` wyznaczający rozwinięcie dziesiętne ułamka zwykłego z dokładnością do `n` cyfr po przecinku. Przykładowa sesja:

```
x = 1/17
n = 15

x = 0.058823529411765
```

Projekt 16 (4.0)

Napisz program `srednia.asm` obliczający średnią z ocen. Pobieranie ocen kończy się po wprowadzeniu liczby 0. W przypadku podania niepoprawnej oceny program ponawia prompt. Przykładowa sesja:

```
1 ocena: 4.5
2 ocena: 3.7
2 ocena: 3
3 ocena: 0

srednia: 3.75
```

Projekt 17 (4.0)

Napisz program `wielomian.asm` obliczający wartość wielomianu w sposób klasyczny. Przykładowa sesja:

```
x = 1.5

n = 2

a0 = 1.5
a1 = 2
a2 = 3.5

w(1.500000) = 12.375000
```

Projekt 18 (4.0)

Napisz program `horner.asm` obliczający wartość wielomianu przy pomocy schematu Hornera. Przykładowa sesja:

```
x = 1.5

n = 2

a0 = 1.5
a1 = 2
a2 = 3.5

w(1.500000) = 12.375000
```

Projekt 19 (4.5)

Napisz programr_sequence.asmwyliczający n-ty wyraz ciągu $\{seq_n\}$metodą dziel i zwyciężaj. Ciąg zdefiniowany jest przez wzór rekurencyjny:

$seq_1 = 3$
$seq_2 = 4$
$seq_n = 0.5*seq_{n-1} + 2*seq_{n-2}$, dla $n > 2$

Przykładowa sesja:

```
n = 4
seq(4) = 12.000000
```

Projekt 20 (4.5)

Napisz program sequence.asm wyliczający n-ty wyraz ciągu $\{seq_n\}$metodą programowania dynamicznego z wykorzystaniem ramki dwuzębnej i trójzębnej. Zaproponuj i przeprowadź testy porównawcze czasu wykonania funkcji napisanych w asemblerze i analogicznych funkcji napisanych w języku C. Ciąg zdefiniowany jest przez wzór rekurencyjny:

$seq_1 = 3$
$seq_2 = 4$
$seq_n = 0.5*seq_{n-1} + 2*seq_{n-2}$, dla $n > 2$

Projekt 21 (4.5)

Napisz program permutacje.asmwyznaczający permutacje zbioru {1, …, n} metodą generowania i testowania.

www.balois.pl/java/rekurencja/permutacje.htm

Przykładowa sesja:

```
n = 3

1 2 3
1 3 2
2 1 3
2 3 1
3 1 2
3 2 1
```

Projekt 22 (5.0)

Napisz programdodawanie.asm realizujący dodawanie pisemne. Interfejs programu musi wyglądać dokładnie tak samo, jak w programie dodawanie.exe na stronie autora. Zakładamy, że dane wejściowe mają postać:

```
Z: a = 0, 1, 2, ...
   b = 0, 1, 2, ...
```

- w jakich przypadkach program może dawać niepoprawne wyniki?

Przykładowa sesja:

```
a = 9237
b = 1267

 1 11
  9237
+ 1267
------
 10504
```

- jaka liczba jest wyświetlana jako pierwsza?

- ile może być maksymalnie przeniesień przy dodawaniu?

- ile wynosi i od czego zależy szerokość słupka dodawania?

Projekt 23 (5.0)

Napisz analogiczny program `dodawanie2.asm` przechowujący przeniesienia w tablicy znaków oraz dokonaj optymalizacji kodu, którą umożliwia to rozwiązanie.

Projekt 24 (5.0)

Napisz program `odejmowanie.asm`realizujący odejmowanie pisemne. Interfejs programu musi wyglądać dokładnie tak samo, jak w programie `odejmowanie.exe` na stronie autora. Zakładamy, że dane wejściowe mają postać:

```
Z: a >= b
   a = 0, 1, 2, ...
   b = 0, 1, 2, ...
```

Przykładowa sesja:

```
a = 3513
b = 605

  215 013
  3 5 1 3
-   6 0 5
---------
  2 9 0 8
```

Projekt 25 (5.0)

Napisz program `mnozenie.asm` realizujący mnożenie pisemne. Interfejs programu musi wyglądać dokładnie tak samo, jak w programie `mnozenie.exe` na stronie autora. Zakładamy, że dane wejściowe mają postać:

```
Z: a = 0, 1, 2, ...
   b = 0, 1, 2, ...
```

- w jakich przypadkach program może dawać niepoprawne wyniki?

Przykładowa sesja:

```
a = 372
b = 57

   131
   251
    372
  *  57
  -----
   2604
+ 1860
-------
  21204
```

Projekt 26 (4.5)

Napisz program `sumowanie.asm` obliczający sumę dwóch dowolnie wielkich liczb nieujemnych. W przypadku podania niepoprawnej liczby program ponawia prompt. Przykładowa sesja:

```
a = zonk
a = 18446744073709551615
b = 18446744073709551615

suma = 36893488147419103230
```

Projekt 27 (4.5)

Napisz program `roznica.asm` obliczający różnicę dwóch dowolnie wielkich liczb nieujemnych. W przypadku podania niepoprawnej liczby program ponawia prompt. Przykładowa sesja:

```
a = zonk
a = 1234
b = 18446744073709551615999

roznica = -18446744073709551614765
```

Projekt 28 (5.0)

Napisz program `iloczyn.asm` obliczający iloczyn dwóch dowolnie wielkich liczb nieujemnych. W przypadku podania niepoprawnej liczby program ponawia prompt. Przykładowa sesja:

```
a = zonk
a = 1234
b = 18446744073709551615999

iloczyn = 22763282186957586694142766
```

Projekt 29 (3.5)

Napisz program `armstrong.asm` wyznaczający wszystkie liczby Armstronga z zakresu od 100 do `n`, dla podanej wartości naturalnej `n` większej od 100. W przypadku podania niepoprawnej wartości `n` program ponawia prompt. Przykładowa sesja:

```
n = 15
n = 1000

liczby: 153 370 371 407
```

Projekt 30 (3.0)

Napisz program `collatz.asm` wyznaczający ciąg Collatza dla podanej liczby naturalnej `n`. W przypadku podania niepoprawnej wartości `n` program ponawia prompt. Przykładowa sesja:

```
n = - 1
n = 11

ciag: 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1
```

Projekt 31 (3.5)

Napisz program `friendly.asm` wyznaczający liczby zaprzyjaźnione w przedziale od 1 do podanej liczby naturalnej `n`. W przypadku podania niepoprawnej wartości `n` program ponawia prompt. Przykładowa sesja:

```
n = - 1
n = 1000

6  6
28 28
220 284
284 220
496 496
```

Projekt 32 (4.0)

Napisz program `fortunat.asm` sprawdzający czy podana liczba naturalna `n` jest liczbą Fortunatego. W przypadku podania niepoprawnej wartości `n` program ponawia prompt. Przykładowa sesja:

```
n = - 1
n = 17

17 jest liczba Fortunatego
```