Pytania i zadania przykładowe do ćwiczeń z „Architektury komputerów”

cz. II, grudzień 2017

1. Poniżej podano fragment programu w języku C.

```
int a, b, * wsk, wynik;
wsk = &b;
a = 21; b = 25;
wynik = roznica(&a, &wsk);
```

Napisać podprogram w asemblerze przystosowany do wywoływania z poziomu języka C, którego prototyp ma postać:

```
int roznica (int * odjemna,
                  int ** odjemnik);
```

Podprogram ten powinien obliczyć różnicę dwóch liczb całkowitych ze znakiem w kodzie U2.

2. Funkcja biblioteczna języka C o prototypie

```
void * malloc(unsigned int k);
```

przydziela `k`-bajtowy obszar pamięci i zwraca adres przydzielonego obszaru. Jeśli wymagany obszar nie może być przydzielony, to funkcja zwraca wartość 0.
Napisać podprogram w asemblerze, przystosowany do wywoływania z poziomu języka C — prototyp tego podprogramu ma postać:

```
int * kopia_tablicy(int tabl[],
            unsigned int n);
```

Podprogram `kopia_tablicy` tworzy nową tablicę o rozmiarach identycznych z oryginalną i zwraca adres nowej tablicy (albo 0, jeśli tablicy nie można było utworzyć).
Do nowej tablicy należy skopiować wszystkie elementy tablicy oryginalnej o wartościach będących liczbami parzystymi. Pozostałe elementy nowej tablicy wypełnić zerami.
Wskazówki:

1. Nową tablicę należy utworzyć poprzez przydzielenie odpowiedniego obszaru pamięci za pomocą funkcji `malloc`.
2. Sprawdzenie czy liczba jest parzysta najłatwiej wykonać poprzez odczytanie najmłodszego bitu liczby (bit nr 0).

3. Podprogram, przystosowany do wywoływania z poziomu języka C, o prototypie:

```
char * komunikat (char * tekst);
```

rezerwuje obszar pamięci (za pomocą funkcji `malloc`) i wpisuje do niego tekst wskazany przez parametr `tekst`, bezpośrednio za którym zostaje dopisany łańcuch znaków `"Błąd."`. Podprogram zwraca adres przydzielonego obszaru. Napisać kod asemblerowy tego podprogramu.
Wskazówka: łańcuch znaków kończy się bajtem o wartości 0.

4. Poniżej podano fragment programu w języku C.

```
int pomiary[7], * wsk;
- - - - - - - - - - - - - - -
wsk = szukaj_elem_min(pomiary, 7);
printf("\nElement minimalny = %d\n",
                          * wsk);
```

Napisać podprogram w asemblerze przystosowany do wywoływania z poziomu języka C, którego prototyp ma postać:

```
int * szukaj_elem_min (
            int tablica[ ], int n);
```

Podprogram ten powinien wyznaczyć najmniejszy element tablicy i zwrócić adres (wskaźnik) tego elementu. Liczbę elementów tablicy określa drugi parametr funkcji.

5. Podprogram, przystosowany do wywoływania z poziomu języka C, o prototypie:

```
void szyfruj (char * tekst);
```

szyfruje każdy bajt obszaru `tekst` poprzez wykonanie operacji XOR, której drugim argumentem jest ciąg 8 bitów. Ciąg ma inną postać dla każdego bajtu i stanowi 8 najmłodszych bitów 32-bitowej liczby losowej. Pierwsza liczba losowa ma wartość 52525252H, a następne obliczane są w poniższy sposób:

a. wyznacza się sumę modulo dwa bitów nr 30 i 31,
b. przesuwa się całą liczbę 32-bitową o jedną pozycję w lewo,
c. na bit nr 0 liczby wprowadza się wcześniej obliczoną sumę modulo dwa.

Napisać kod podprogramu `szyfruj` w asemblerze.

6. Napisać podprogram w asemblerze obliczający wartość funkcji kwadrat metodą rekurencyjną korzystając z zależności:

$$a^2 = (a-2)^2 + 4*a - 4 \quad \text{dla } a > 1$$
$$a^2 = 1 \quad \text{dla } a = 1$$
$$a^2 = 0 \quad \text{dla } a = 0$$

przy czym argument a jest liczbą całkowitą 32-bitową zawartą w przedziale <1, 65535>.
Podprogram powinien być przystosowany do wywoływania z poziomu języka C, a jego prototyp ma postać:

```
unsigned int kwadrat (
                    unsigned int a);
```

W podprogramie nie można używać rozkazów mnożenia i rozkazów przesunięć.

7. W pewnym programie używana jest funkcja `iteracja`, której prototyp ma postać:

```
unsigned char iteracja(unsigned char a);
```

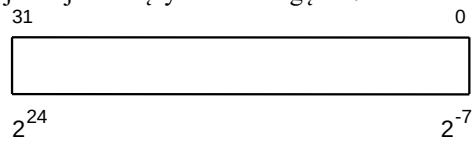
Podać wartość zwracaną przez funkcję `iteracja` w poniższym fragmencie programu w języku C

```
w = iteracja(32);
```

Kod funkcji `iteracja` zapisany w asemblerze ma postać

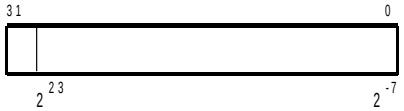
```
_iteracja     PROC
           push    ebp
           mov     ebp, esp
           mov     al, [ebp+8]
           sal     al, 1
; SAL wykonuje przesuniecie logiczne
; w lewo
           jc      zakoncz
           inc     al
           push    eax
           call    _iteracja
           add     esp, 4
           pop     ebp
           ret

zakoncz:   rcr     al, 1
; rozkaz RCR wykonuje przesunięcie
; cykliczne w prawo przez CF
           pop     ebp
           ret
_iteracja     ENDP
```

8. W programie asemblerowym zdefiniowano format 32-bitowych liczb mieszanych bez znaku, przyjmując, że najmniej znaczący bit ma wagę 2^{-7}.



Napisać fragment programu w asemblerze, który wpisze 1 do znacznika CF (rozkaz STC) jeśli część całkowita liczby zawartej w rejestrze EBX jest różna od zera; w przeciwnym razie CF powinien zostać wyzerowany (rozkaz CLC).

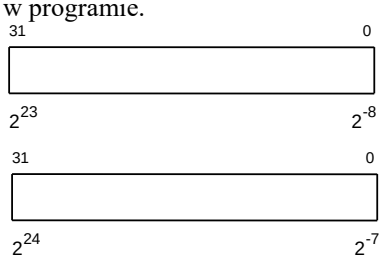
9. W programie zdefiniowano format 32-bitowych liczb mieszanych w kodzie U2 przyjmując, że najmniej znaczący bit ma wagę 2^{-7}.



Przyjmując, że liczba w podanym formacie znajduje się w rejestrze EDX, napisać fragment programu, który zaokrągli tę liczbę do najbliższej liczby całkowitej. Wynik należy pozostawić również w rejestrze EDX w podanym formacie.

Wskazówka: zbadać stan bitu o wadze 2^{-1}.

10. Na poniższym rysunku pokazane są dwa formaty 32-bitowych liczb stałoprzecinkowych bez znaku używane w programie.



Zakładając, że w rejestrze ESI znajduje się liczba zakodowana wg pierwszego formatu (najmniej znaczący bit ma wagę 2^{-8}), a w rejestrze EDI liczba zakodowana wg drugiego formatu (najmniej znaczący bit ma wagę 2^{-7}), napisać fragmentu, który porówna obie liczby. Jeśli liczba w rejestrze ESI jest większa od liczby w rejestrze EDI, to do CF należy wpisać 1, w przeciwnym razie 0 (rozkazy STC/CLC).

11. Podać reprezentację liczby –0.25 w formacie zmiennoprzecinkowym *double*. Wskazówka: pole mantysy w formacie w formacie *double* zajmuje 52 bity.

12. Poniżej podano reprezentację binarną liczby –8.25 w postaci 32-bitowej liczby zmiennoprzecinkowej (*float*). Uzupełnić brakujące bity mantysy w tej reprezentacji.

	10000010					00000000000000000000

13. Poniżej podano reprezentację binarną dwóch 32-bitowych liczb binarnych zmiennoprzecinkowych (format *float*) **x** i **y**. Ile wynosi iloraz **x/y** tych liczb ?

x	0	10000001	00000000000000000000000
y	1	10000000	00000000000000000000000

14. W pamięci komputera, począwszy od adresu podanego w rejestrze ESI znajduje się 32-bitowa liczba zmiennoprzecinkowa w formacie *float*. Napisać fragment programu, który przekształci tę liczbę na 64-bitowy format *double* i uzyskany rezultat wpisze do pamięci począwszy od adresu podanego w rejestrze EDI. W omawianym fragmencie nie można używać rozkazów koprocesora arytmetycznego.

15. Poniższy program w języku C wczytuje promień `r` koła z klawiatury, oblicza pole koła i wyświetla na ekranie wynik obliczenia.
Obliczenie pola koła wykonuje podprogram zakodowany w asemblerze, przystosowany do wywoływania z poziomu języka C, którego prototyp na poziomie języka C ma postać:

```
void pole_kola (float * pr);
```

Argument `pr` jest adresem (wskaźnikiem) zmiennej, w której przechowywany jest promień `r` koła.
Napisać podprogram w asemblerze dokonujący opisanego obliczenia i uruchomić program składający się z plików źródłowych w języku C i w asemblerze.

```
#include <stdio.h>
void pole_kola (float * pr);
int main()
{
  float k;
  printf("\nProszę podać promień: ");
  scanf_s("%f", &k);
  pole_kola (&k);
  printf("\nPole koła wynosi %f\n",
                                    k);
  return 0;
}
```

16. Poniżej podano kod (niepełny) podprogramu w asemblerze, który zwiększa o 1 liczbę zmiennoprzecinkową w formacie `double`. Podprogram przystosowany jest do wywoływania z poziomu języka C, a jego prototyp ma postać:

```
double plus_jeden (double x);
```

Nie używając rozkazów koprocesora, uzupełnić brakujący fragment podprogramu, przy założeniu, że liczba `x` jest większa od 1, a zawartość pola wykładnika (w formacie `double`) należy do przedziału <1023, 1075>. Uwaga: po wykonaniu dodawania przeprowadzić normalizację liczby.

```
_plus_jeden        PROC
           push    ebp
           mov     ebp, esp
           push    ebx
           push    esi
           push    edi
; odczytanie liczby
; w formacie double
           mov     eax, [ebp+8]
           mov     edx, [ebp+12]
; wpisanie 1 na pozycji o wadze 2^0
; mantysy do EDI:ESI
           mov     esi, 0
           mov     edi, 00100000H
; wyodrębnienie pola
; wykładnika (11-bitowy)
; bit znaku liczby z założenia = 0
           mov     ebx, edx
           shr     ebx, 20
; obliczenie pierwotnego wykładnika
; potęgi
           sub     ebx, 1023
; zerowanie wykładnika i bitu znaku
           and     edx, 000FFFFFH
; dopisanie niejawnej jedynki
           or      edx, 00100000H
_  _  _  _  _  _  _  _  _  _  _  _  _  _
_  _  _  _  _  _  _  _  _  _  _  _  _  _
; załadowanie obliczonej wartości z
; EDX:EAX na wierzchołek stosu
; koprocesora
           push    edx
           push    eax
           fld     qword PTR [esp]
           add     esp, 8
           pop     edi
           pop     esi
           pop     ebx
           pop     ebp
           ret
_plus_jeden        ENDP
```

17. Niżej opisany podprogram, przystosowany do wywoływania z poziomu języka C, posiada prototyp:

```
float  avg_wd (int n, void *
    tablica, void * wagi)
```

Napisz funkcję `avg_wd` w asemblerze z wykorzystaniem rozkazów koprocesora. Zakładając, że `tablica` i `wagi` są n-elementowymi tablicami liczb zmiennoprzecinkowych typu *float*, należy obliczyć średnią ważoną wszystkich wyrazów tablicy. Przed zakończeniem podprogramu wszelkie wyniki pośrednie znajdujące się na stosie rejestrów koprocesora powinny zostać usunięte.

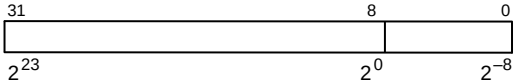
18. Napisać podprogram w asemblerze, przystosowany do wywoływania poziomu języka C, o prototypie:

```
unsigned int NWD(unsigned int a,
                 unsigned int b);
```

Podprogram ten powinien wyznaczyć wartość największego wspólnego podzielnika dla dwóch liczb `a` i `b`. Obliczenie należy przeprowadzić metodą rekurencyjną wg algorytmu przedstawionego poniżej.

```
unsigned int NWD(unsigned int a,
unsigned int b)
{
  unsigned int wyn;
  if (a == b) wyn = a;
  else if (a > b) wyn = NWD (a - b, b);
  else wyn = NWD (a, b - a);
  return wyn;
}
```

19. W programie zdefiniowano 32-bitowy typ `MIESZ32`, w którym kodowane są liczby stałoprzecinkowe bez znaku zawierające część całkowitą i ułamkową, tak jak pokazano na rysunku.



Napisać podprogram w asemblerze przystosowany do wywoływania z poziomu języka C, który zamieni liczbę `p` w podanym formacie na 32-bitową liczbę zmiennoprzecinkową w formacie *float*. Prototyp podprogramu na poziomie języka C ma postać:

```
float  miesz2float (MIESZ32  p);
```

Zalecenia i wskazówki:

a. Podprogram powinien przeprowadzić konwersję posługując się rozkazami procesora (nie koprocesora). Wartości typu `MIESZ32` przekazywane są przez stos wg takich samych reguł jak wartości typu `int`. Pominąć przypadek liczby 0.
b. Odszukać położenie najstarszego bitu o wartości 1 w liczbie `p`, i następnie przesunąć tę liczbę w prawo lub w lewo o odpowiednią liczbę bitów, dostosowując ją do formatu *float*.
c. Wyznaczyć także wartość pola wykładnika i wpisać ją na odpowiednie bity. Wynik w postaci liczby typu *float* umieścić na wierzchołku stosu koprocesora.

20. Napisać podprogram w asemblerze przystosowany do wywoływania z poziomu języka C, którego prototyp ma postać:

```
unsigned int liczba_procesorow();
```

który zwróci jako wyjście liczbę procesorów w systemie. Napisać przykładowy program w języku C demonstrujący użycie funkcji.
Wskazówki:

1. użyj funkcji GetSystemInfo (opis na msdn.microsoft.com).

21. Napisać podprogram w asemblerze przystosowany do wywoływania z poziomu języka C, o podanym niżej prototypie

```
float  float_razy_float (float a, float b);
```

Podprogram ten powinien wyznaczyć iloczyn dwóch liczb zmiennoprzecinkowych typu `float`. Obliczenie wykonać bez wykorzystania rozkazów koprocesora arytmetycznego (z wyjątkiem `FLD`). Zakładamy, że obie liczby są dodatnie, a ich iloczyn da się przedstawić w postaci liczby typu `float`.

Wskazówki:

a. Zadanie wymaga wyznaczenia iloczynu mantys obu liczb i dodania wykładników (wykładnik jest przesunięty o 127).
b. W wyniku mnożenia dwóch liczb binarnych, w których najmłodszy bit ma wagę 2^{-23} uzyskujemy iloczyn, w którym najmłodszy bit ma wagę 2^{-46}. Należy dostosować format tego iloczynu do wymagań formatu `float`.
c. Jeśli wartość iloczynu jest większa lub równa od 2, to należy go znormalizować i odpowiednio skorygować wykładnik.

22. Napisać podprogram w asemblerze, przystosowany do wywoływania z poziomu języka C, o prototypie:

```
unsigned __int64 sortowanie (unsigned
   __int64 * tabl, unsigned int n);
```

Podprogram powinien posortować zawartość `n`-elementowej tablicy `tabl` w porządku rosnącym i zwrócić wartość największego elementu. Elementami tablicy są liczby 64-bitowe bez znaku, a program wykonywany jest w trybie 32-bitowym.
Wskazówka: 64-bitowe wartości funkcji w trybie 32-bitowym przekazywane są przez rejestry EDX:EAX.

23. Napisać podprogram w asemblerze, przystosowany do wywoływania z poziomu języka C, o prototypie:

```
wchar_t * ASCII_na_UTF16(char * znaki,
                         int n);
```

Podprogram powinien przekształcić ciąg `n` znaków w kodzie ASCII (kody < 128) na odpowiadający mu ciąg znaków w standardzie UTF16. Uzyskane znaki należy wpisać do obszaru pamięci przydzielonego za pomocą funkcji `malloc`. Na końcu ciągu znaków należy umieścić liczbę 0 (16-bitową). Adres przydzielonego obszaru należy zwrócić jako wartość funkcji. Typ `wchar_t` reprezentuje znaki 16-bitowe używane w Unikodzie. Funkcja biblioteczna języka C `malloc(k)` przydziela k-bajtowy obszar pamięci i zwraca adres przydzielonego obszaru. Zawartość przydzielonej pamięci jest nieokreślona.