Języki formalne i metody kompilacji
Laboratorium nr 11

Stefan Sokołowski

Zadanie 1:

Zakładamy, że na każdą zmienną całkowitą przewidziane jest rezerwowanie 4 bajtów pamięci a na zmienną rzeczywistą 8 bajtów. Zakładamy też, że dane składające się na zmienną tablicową `tab` o deklaracji

```
typedef struct {
  int a;
  double b;
} QQ;
QQ tab[20][30];
```

zajmują spójny obszar pamięci zaczynający się od bajtu o adresie 5 000. Które bajty zajmą zmienne wymienione obok?

zmienna	pierwszy bajt	ostatni bajt
tab[0][0]		
tab[5][10]		
tab[0][0].a		
tab[5][10].b		

Zadanie 2:

Zakładamy, że na każdą zmienną całkowitą przewidziane jest rezerwowanie 4 bajtów pamięci, na zmienną rzeczywistą 8 bajtów a na zmienną znakową 1 bajt. Zakładamy też, że dane składające się na zmienną tablicową `xx` o deklaracji

```
typedef struct {
  double a;
  char b[10][15];
  int c[12];
} ABC;
ABC xx[10][100][20];
```

zajmują spójny obszar pamięci zaczynający się od bajtu o adresie 1 000. Które bajty zajmą zmienne wymienione obok?

zmienna	pierwszy bajt	ostatni bajt
xx[2][3][4]		
xx[2][3][4].b[2][3]		
xx[2][3][4].c[10]		

Zadanie 3:

Zakładamy, że na każdą zmienną rzeczywistą przewidziane jest rezerwowanie 8 bajtów pamięci. Zakładamy też, że dane składające się na zmienną tablicową `tab` o deklaracji

```
typedef struct {
  double re; double im;
} zespol;
zespol tab[n][m];
```

zajmują spójny obszar pamięci zaczynający się od bajtu o jakimś adresie `adr`. Które bajty, w zależności od wartości `n`, `m`, `p` i `adr`, zajmą zmienne wymienione obok?

zmienna	pierwszy bajt	ostatni bajt
$\texttt{tab}[2]$		
$\texttt{tab}[2][\texttt{m}-3].\texttt{re}$		
$\texttt{tab}[\texttt{p}][2*\texttt{p}].\texttt{im}$		

Proszę założyć, że

- $\texttt{n} > 2\,,\ \texttt{m} \geqslant 3\,,$
- $0 \leqslant \texttt{p} < \texttt{n}\,,\ 2\cdot\texttt{p} < \texttt{m}\,.$

Zadanie 4 (**domowe na punkty**)**:**

Obok podana jest **rekursywna** funkcja w C, która czerpie liczbę zapisaną w miejscu wskazywanym przez zmienną `n`, oblicza jej silnię i umieszcza wynik w miejscu wskazywanym przez zmienną `s`.

```
void silnia(int* n, int* s) {
  int* n1 = (int*)malloc(sizeof(int));
  int* s1 = (int*)malloc(sizeof(int));
  if (*n == 0) *s = 1;
  else {
    *n1 = *n-1; silnia(n1, s1); *s = *s1 * *n;
  }
}
```

Program w języku „wewnętrznym” podany poniżej robi to samo; a jego zasadniczą funkcję (od komórki 11 do komórki 32) można uznać za implementację powyższej silni. Rekursja realizowana jest przy pomocy stosu, który zaczyna się w komórce 37 i na który przy przejściu na kolejny poziom rekursji odkładane są trzy komórki:

- miejsce na daną liczbę,
- miejsce na wynik,
- miejsce na adres powrotu z danego poziomu rekursji.

Tak więc komórki 37–39 obsługują zerowy poziom rekursji, komórki 40–42 — pierwszy poziom, komórki 43–45 — drugi poziom, itd. Dostęp do danych na stosie z aktualnego poziomu jest realizowany przez modyfikację adresową komórką 35. Wywołania obliczenia silni na aktualnym poziomie rekursji dokonuje się przez skok pod adres 11.

```
 0: 7                    # tu nalezy wpisac dana n
 1: 0                    # tu program wpisze wynik s = n!
 2: loada 0            # POCZATEK PROGRAMU GLOWNEGO
 3: store 37             # dana n do funkcji
 4: loadn 7
 5: store 39             # adres powrotu do funkcji
 6: goto 11              # wywolanie funkcji
 7: loada 38
 8: store 1              # wynik funkcji do programu
 9: stop               # KONIEC PROGRAMU GLOWNEGO
10: 0#-----------
11: loada 37+[35]      # POCZATEK FUNKCJI SILNIA
12: zergoto 28
13: incr 35             # jesli dana n != 0
14: incr 35
15: incr 35             # na wyzszy poziom rekursji
16: store 37+[35]
17: decr 37+[35]        # wlozyc n-1 na stos
18: loadn 21
19: store 39+[35]       # adres powrotu
20: goto 11             # wywolanie rekursywne
21: loada 38+[35]       # wynik czyli (n-1)!
22: decr 35
23: decr 35
24: decr 35             # powrot z wywolania rekursywnego
25: mul 37+[35]
26: store 38+[35]       # wynik s = (n-1)!*n
27: goto 30
28: loadn 1             # jesli dana n = 0
29: store 38+[35]       # wynik s = 1
30: loada 39+[35]
31: store 33            # przygotowanie adresu powrotu
32: goto 0+[33]        # KONIEC FUNKCJI SILNIA
33: 0                   # lokalny adres powrotu z funkcji
34: 1                   # stala 1
35: 0                   # trzykrotnosc numeru poziomu rekursji
36: 0#----------- STOS:
37: 0                   # poziom 0: dana n
38: 0                   # poziom 0: wynik s = n!
39: 0                   # poziom 0: adres powrotu
END
```

Proszę w podobny sposób skonstruować przekład na język „wewnętrzny” odpowiednika funkcji zdefiniowanej rekursywnie przez

$$f(n) = \begin{cases} 0 & \text{jeśli } n = 0 \\ f(n-1) + n^2 & \text{jeśli } n \geqslant 1 \end{cases}$$

Uwaga: równoważne obliczenie łatwo jest zorganizować bez rekursji i (wobec tego) bez operacji na stosie; jednak sensem tego zadania jest właśnie organizacja rekursji, więc **tego** proszę nie upraszczać.

```
void f(int* n, int* s) {
  int* n1 =
    (int*)malloc(sizeof(int));
  int* s1 =
    (int*)malloc(sizeof(int));
  if (*n == 0) *s = 0;
  else {
    *n1 = *n-1; f(n1, s1);
    *s = *s1 + (*n)*(*n);
  }
}
```

Uwaga:

Czas na przysłanie (na `s.sokolowski@ans-elblag.pl`): 2 tygodnie od zajęć z lab. 11 — potem już nie przyjmuję rozwiązań. Liczba punktów zależy od oryginalności i elegancji rozwiązania.

Zadanie 5 (**domowe specjalne**)**:**

Na wzór programu z zad. 4 proszę napisać program w języku „wewnętrznym” rekursywnie **obliczający** n**-tą liczbę Fibonacciego**:

$$Fib(n) = \begin{cases} 0 & \text{jeśli } n = 0 \\ 1 & \text{jeśli } n = 1 \\ Fib(n-1) + Fib(n-2) & \text{jeśli } n \geqslant 2 \end{cases}$$

Uwaga:

Pierwszej osobie, która przyśle mi (na `s.sokolowski@ans-elblag.pl`) poprawne rozwiązanie tego zadania, przysługują punkty do zaliczenia laboratorium. Liczba tych punktów zależy od oryginalności i elegancji rozwiązania.
