

# Architektury systemów komputerowych

Lista 8

$x_8 = 8$  (minimum na bdb)

1. W tym zadaniu porównamy listy rozkazów o różnych liczbach argumentów.

(a) Masz do dyspozycji następujące rozkazy:

PUSH A    umieść zawartość rejestru A na stosie  
POP A     umieść zawartość wierzchołka stosu w rejestrze A  
ADD       zdejmij ze stosu dwie wartości, dodaj je do siebie i wynik umieść na stosie  
SUB       zdejmij ze stosu dwie wartości, odejmij je od siebie i wynik umieść na stosie  
MUL       zdejmij ze stosu dwie wartości, pomnóż je przez siebie i wynik umieść na stosie

Napisz program obliczający wartość wyrażenia:  $(XY - Z)(X + ZY)$  i umieszczający wynik w rejestrze W. Zakładamy, że  $X, Y, Z$  reprezentują w tym wyrażeniu zawartość rejestrów  $X, Y, Z$ . Chcemy, aby zawartość rejestrów  $X, Y, Z$  pozostała nienaruszona. Jeśli chcesz, możesz skorzystać z pomocniczych rejestrów  $A_1, \dots, A_{10}$ .

(b) Napisz program obliczający wartość wyrażenia z zadania 1a (przy takich samych założeniach o rejestrach  $X, Y, Z, W, A_1, \dots, A_{10}$ ) na maszynie o liście rozkazów podanej poniżej. Specjalny rejestr ACC jest akumulatorem.

LOAD M    załaduj zawartość M do akumulatora  
STORE M   zapisz zawartość akumulatora do M  
ADD M     dodaj zawartość M do akumulatora  
SUB M     odejmij zawartość M od akumulatora  
MUL M     pomnóż akumulator przez zawartość M

(c) Jak w zadaniu poprzednim, tym razem jednak nie ma rejestru ACC i dysponujemy rozkazami dwuargumentowymi:

MOVE X Y   do X zapisz zawartość Y  
ADD X Y    dodaj zawartość X do zawartości Y, wynik zapisz w X  
SUB X Y    odejmij zawartość Y od zawartości X, wynik zapisz w X  
MUL X Y    pomnóż zawartość Y przez zawartość X, wynik zapisz w X

(d) Jak w zadaniu poprzednim, tym razem jednak dysponujemy rozkazami trzyargumentowymi:

MOVE X Y   do X zapisz zawartość Y  
ADD X Y Z   dodaj zawartość Y do zawartości Z, wynik zapisz w X  
SUB X Y Z   odejmij zawartość Z od zawartości Y, wynik zapisz w X  
MUL X Y Z   pomnóż zawartość Y do zawartości Z, wynik zapisz w X

2. Napisz program na maszynie z zadania 1a wyznaczający minimum z zawartości trzech rejestrów  $X, Y, Z$  i umieszczający wynik w rejestrze W. Dodajemy dodatkowy rozkaz JMPGTZ E, który przeskakuje do instrukcji poprzedzonej etykietą E gdy na szczycie stosu jest liczba dodatnia (w przeciwnym wypadku przechodzi do następnej instrukcji).

3. Rozważmy hipotetyczną maszynę, której lista rozkazów składa się tylko z jednego, trzyargumentowego rozkazu SBN (*subtract and branch if negative* - odejmij i skocz jeśli wynik ujemny). Rozkaz SBN A B C odejmuje zawartość komórki pamięci A od zawartości komórki pamięci B i wynik umieszcza w A. Następnie, jeśli wynik był ujemny, skacze do instrukcji o adresie C, jeśli wynik był nieujemny – przechodzi do następnej instrukcji.

W języku wyższego poziomu SBN A B C odpowiada (przyjmujemy, że PC to zmienna odpowiadająca licznikowi rozkazów) program:

```
mem[A] := mem[A] - mem[B];  
if (mem[A] < 0) then PC = C;  
else PC = PC + 1
```

Jedynym rejestrem rozważanej maszyny jest licznik rozkazów PC. Zobaczymy, że nasza maszyna jest w rzeczywistości w stanie wykonywać równie złożone zadania, co maszyny z bogatszymi listami rozkazów. Oto przykład programu kopiującego zawartość komórki pamięci  $A$  do komórki pamięci  $B$  (liczby po lewej stronie to adresy, w których umieszczamy rozkazy,  $T$ ,  $A$ ,  $B$  – symboliczne adresy komórek pamięci, napisy po średnikach – komentarze):

```
100:      SBN    T, T, 101    ; Mem[T]:=0
101:      SBN    T, A, 102    ; Mem[T]:=Mem[T]-Mem[A]  ( = 0-mem[A] = -Mem[A])
102:      SBN    B, B, 103    ; Mem[B]:=0
103:      SBN    B, T, 104    ; Mem[B]:=-Mem[T]  ( = +Mem[A])
```

Napisz program na naszą hipotetyczną maszynę, który dodaje zawartość komórki pamięci  $B$  do zawartości komórki  $A$  i wynik umieszcza w  $A$ . Zawartość komórki  $B$  nie może zostać zmieniona, ale możesz użyć pomocniczych komórek pamięci  $T_1, \dots, T_n$ .

4. Napisz program na maszynę z zadania 3 mnożący zawartość komórki pamięci  $A$  przez siebie i umieszczający wynik w  $C$ . Możesz założyć, że zawartość  $A$  jest dodatnia, a w komórce pamięci  $X$  jest zapisana wartość 1. Program może zmieniać zawartość pamięci (również komórki  $A$ ), chcemy jedynie by na koniec w  $C$  był kwadrat początkowej zawartości  $A$ .

**Wskazówka:** zorganizuj pętlę, która  $a$  razy doda do siebie  $a$ , gdzie  $a$  jest zawartością  $A$ ; przyda Ci się rozwiązanie zadania 3.

5. Zasymuluj poniższy fragment kodu w assemblerze procesora MIPS

```
g = -f + h + B[1];
f = A[B[g]+1];
```

Zakładamy, że zmienne  $f, g, h$  przechowywane są odpowiednio w rejestrach  $\$s0, \$s1, \$s2$ , a adresy elementów  $A[0]$  i  $B[0]$  w rejestrach  $\$s3$  i  $\$s4$ .

6. Zasymuluj poniższe fragmenty kodu w języku assemblera MIPS:

```
(a) if f > 1 then
    g = f + f;
    f = 0;
endif;
    g = g + 1;

(b)
f = 1;
while f < 10 do
    f = f + 1;
endwhile

(c) g = 0;
for f = 1 to 10 do
    g := g + f;
```

7. (a) Przetłumacz następujące rozkazy assemblera procesora MIPS na kod maszynowy:

- add  $\$t0, \$t0, \$zero$ ,
- lw  $\$t1, 4(\$s3)$

- (b) W kodzie maszynowym pewne rozkazy wyglądają następująco (przedstawiamy je tutaj szesnastkowo):  $0xAE0BFFFC$  oraz  $0x8D08FFC0$ . Jak te rozkazy wyglądają na poziomie assemblera?

8. Rozważmy następujący program w assemblerze procesora MIPS:

```
    addi $t1, $s0, 400
LOOP: lw  $s1, 0($s0)
    add  $s2, $s2, $s1
    lw  $s1, 4($s0)
    add  $s2, $s2, $s1
    addi $s0, $s0, 8
    bne $t1, $s0, LOOP
```

- (a) Jaka będzie całkowita liczba wykonanych rozkazów?
  - (b) Przetłumacz powyższy kod na język C, zakładając, że `$t1` przechowuje zmienną `i`, `$s2` zmienną `result`, a `$s0` adres tablicy `MemArray`.
9. W assemblerze procesora MIPS napisz program, który zlicza ile jedynek znajduje się w bitowej reprezentacji rejestru `$s0`.
  10. W assemblerze procesora MIPS napisz program wyznaczający największy wspólny dzielnik oraz najmniejszą wspólną wielokrotność dwóch liczb naturalnych (załóż, że argumenty znajdują się w rejestrach `$a0` oraz `$a1`, a wyniki mają się znaleźć w rejestrach `$v0` oraz `$v1`).

*Emanuel Kieroński*