Architektura systemów komputerowych
WI ZUT

Zadania z asemblera 2024
wersja 1028

Zadanie 1: Napisz kod w asemblerze odpowiadający poniższemu fragmentowi kodu w C.

Zmienna ***a*** 8-bitowa, ***y*** 16-bitowa.
Można wykorzystywać tylko instrukcje z listy poniżej.

Wersja A :

```
MOV  dest, source
SHL   dest, quantity
SHR   dest, quantity
ADD  dest, arg
JZ      etykieta
JNZ    etykieta
POP    dest
PUSH source
```

```
y = a * 20;
```

Wersja B :

```
MOV  dest, source
SHL   dest, quantity
SHR   dest, quantity
SUB  dest, arg
JZ      etykieta
JNZ    etykieta
POP    dest
PUSH source
```

Wersja C :

```
MOV  dest, source
ADD  dest, arg
JZ      etykieta
JNZ    etykieta
POP    dest
PUSH source
```

Zadanie 2: Napisz kod w asemblerze odpowiadający poniższemu fragmentowi kodu w C.

Zmienna ***a*** i ***y*** 8-bitowe.

Można wykorzystywać tylko instrukcje z listy poniżej.

```
if( a == 0x32 )
    y = 0;
else
    y = 1;
```

Wersja A :

```
MOV  dest, source
SHL  dest, quantity
SHR  dest, quantity
SUB  dest, arg
JZ   etykieta
JNZ  etykieta
JMP  etykieta
```

Wersja B :

```
MOV  dest, source
SHL  dest, quantity
SHR  dest, quantity
CMP  dest, arg
JZ   etykieta
JMP  etykieta
POP  dest
PUSH source
```

Wersja C :

```
MOV  dest, source
SHL  dest, quantity
SHR  dest, quantity
ADD  dest, arg
JNZ  etykieta
JMP  etykieta
POP  dest
PUSH source
```

Zadanie 3: Napisz kod w asemblerze odpowiadający poniższemu fragmentowi kodu w C.

Zmienna ***a*** i ***y*** 8-bitowe, ***y*** typu bool.
Można wykorzystywać tylko instrukcje z listy poniżej.

Wersja A:

```
MOV  dest, source
SHL  dest, quantity
SHR  dest, quantity
SUB  dest, arg
ADD  dest, arg
XOR  dest, arg
AND  dest, arg
OR   dest, arg
```

```
if( a == 0x32 )
    y = false;
else
    y = true;
```

Zadanie 4: Napisz kod w asemblerze odpowiadający poniższemu fragmentowi kodu w C.

Zmienna ***i*** oraz ***y*** 32-bitowe.
Można wykorzystywać tylko instrukcje z listy poniżej.

Wersja A:

```
MOV  dest, source
SUB  dest, arg
ADD  dest, arg
JZ       etykieta
JNZ    etykieta
JMP    etykieta
XOR  dest, arg
```

```
for( int i=0; i!=5; i++ ) y += i;
```

Wersja B:

```
MOV  dest, source
CMP  dest, arg
ADD  dest, arg
JZ       etykieta
JNZ    etykieta
JMP    etykieta
XOR  dest, arg
```

Wersja C:

```
for( int i=5; i!=0; i-- ) y -= i;
```

Zadanie 5: Napisz kod w asemblerze odpowiadający poniższemu fragmentowi kodu w C.

Zmienna ***y*** 32-bitowa.
Można wykorzystywać tylko instrukcje z listy poniżej.

Wersja A:

```
MOV  dest, source
SUB  dest, arg
ADD  dest, arg
JZ       etykieta
JNZ    etykieta
JMP    etykieta
XOR  dest, arg
```

```
while( y!=5 ) y += 1;
```

Wersja B:

```
MOV  dest, source
CMP  dest, arg
ADD  dest, arg
JZ       etykieta
JNZ    etykieta
JMP    etykieta
XOR  dest, arg
```

Wersja C:

```
do{ y += 1; } while( y!=5 );
```

Wersja D:

```
MOV  dest, source
ADD  dest, arg
JZ       etykieta
JNZ    etykieta
JMP    etykieta
XOR  dest, arg
```

Zadanie 6: Napisz kod w C oraz w asemblerze zliczający liczbę zer w bajcie ***a***.

Zmienna ***a, y*** 8-bitowa.

Można wykorzystywać tylko instrukcje z listy poniżej.

Wersja A:

MOV dest, source
SHL dest, quantity
SHR dest, quantity
AND dest, arg
OR dest, arg
XOR dest, arg
ADD dest, arg
SUB dest, arg

Wersja B: ***PĘTLA***

MOV dest, source
SHL dest, quantity
SHR dest, quantity
AND dest, arg
OR dest, arg
XOR dest, arg
ADD dest, arg
SUB dest, arg
JZ etykieta
JNZ etykieta
JMP etykieta

Np. dla ***a*** == 0x00 ***y*** == 8
Np. dla ***a*** == 0x80 ***y*** == 7
Np. dla ***a*** == 0x12 ***y*** == 6
Np. dla ***a*** == 0x13 ***y*** == 5

Wersja C:

MOV dest, source
SHL dest, quantity
SHR dest, quantity
ADD dest, arg
SUB dest, arg
AND dest, arg
SAR dest, quantity

Zadanie 7: Napisz kod w C oraz w asemblerze określający parzystość liczby jedynek.

Zmienna ***a*** i ***y*** 8-bitowe.
Kod w asemblerze może wykorzystywać tylko instrukcje z listy poniżej.

Wersja A:

MOV dest, source
SHL dest, quantity
SHR dest, quantity
AND dest, arg
OR dest, arg
XOR dest, arg
ADD dest, arg
SUB dest, arg

Wersja B: ***PĘTLA***

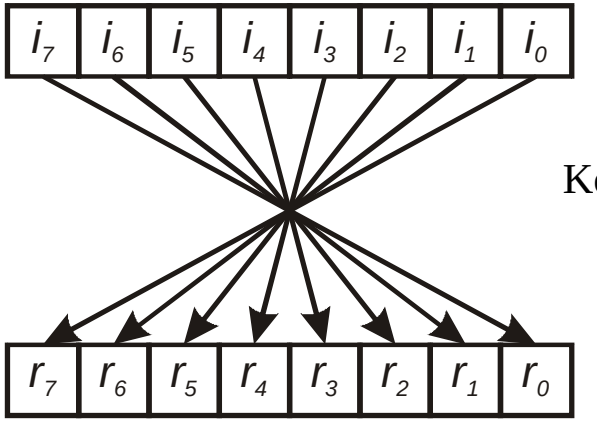
MOV dest, source
SHL dest, quantity
SHR dest, quantity
AND dest, arg
OR dest, arg
XOR dest, arg
ADD dest, arg
SUB dest, arg
JZ etykieta
JNZ etykieta
JMP etykieta

Np. dla ***a*** == 0x00 ***y*** == 1
Np. dla ***a*** == 0x80 ***y*** == 0
Np. dla ***a*** == 0x12 ***y*** == 1
Np. dla ***a*** == 0x13 ***y*** == 0

Wersja C:

MOV dest, source
SHL dest, quantity
SHR dest, quantity
ADD dest, arg
SUB dest, arg
AND dest, arg
SAR dest, quantity

Zadanie 8: Napisz kod w C oraz w asemblerze zamieniający kolejność bitów w bajcie.
Najmłodszy bit zmiennej ***a*** ma być najstarszym zmiennej ***r***, itp., zgodnie z rysunkiem poniżej.



Np. dla ***a*** == 0x80 wynik ***r*** == 0x01.
Np. dla ***a*** == 0x12 wynik ***r*** == 0x48.
Np. dla ***a*** == 0x48 wynik ***r*** == 0x12.

Kod w asemblerze może wykorzystywać tylko instrukcje z listy poniżej:

Wersja A:
MOV dest, source
SHL dest, quantity
SHR dest, quantity
OR dest, arg
AND dest, arg
XOR dest, arg

Wersja B:
MOV dest, source
SHL dest, quantity
SHR dest, quantity
OR dest, arg
TEST dest, arg
XOR dest, arg

Wersja C: ***PĘTLA***
MOV dest, source
SHL dest, quantity
SHR dest, quantity
OR dest, arg
TEST dest, arg
XOR dest, arg
JZ etykieta
JNZ etykieta
JMP etykieta

Zadanie 9: Napisz kod w C oraz asemblerze odpowiadający poniższemu fragmentowi kodu w C (BEZ UŻYCIA MNOŻENIA).

Zmienna ***a*** i ***b*** 8-bitowe, zmienna ***y*** 16-bitowa.
W asemblerze można wykorzystywać tylko instrukcje z listy poniżej.

```
y = a * b;
```

Wersja A:

```
MOV  dest, source
SHL  dest, quantity
SHR  dest, quantity
ADD  dest, arg
AND  dest, arg
JZ   etykieta
JNZ  etykieta
```

Wersja B:

```
MOV  dest, source
SHL  dest, quantity
SHR  dest, quantity
ADD  dest, arg
TEST dest, arg
JZ   etykieta
JNZ  etykieta
```

Wersja C: ***PĘTLA***

```
MOV  dest, source
SHL  dest, quantity
SHR  dest, quantity
ADD  dest, arg
TEST dest, arg
AND  dest, arg
JZ   etykieta
JNZ  etykieta
JMP  etykieta
```

Wersja D: trudne

```
MOV  dest, source
SHL  dest, quantity
SHR  dest, quantity
ADD  dest, arg
AND  dest, arg
```

Zadanie 10: Napisz kod w C oraz asemblerze odpowiadający poniższemu fragmentowi kodu w C (BEZ UŻYCIA MNOŻENIA).

Zmienna ***a*** oraz ***y*** jest liczbą 16-bitową.
W asemblerze można wykorzystywać tylko instrukcje z listy poniżej.

```
MOV  dest, source
SHL   dest, quantity
SHR   dest, quantity
ADD  dest, arg
SUB  dest, arg
TEST dest, arg
```

```
y = a *  1.125;
```

Zadanie 11: Napisz kod w asemblerze odpowiadający poniższemu fragmentowi kodu w C z wykorzystaniem FLAGI C.

Zmienna ***a*** i ***y*** 8-bitowe.
Można wykorzystywać tylko instrukcje z listy poniżej.
W x86 SUB neguję flagę C !!!

Wersja A:

```
MOV  dest, source
SHL  dest, quantity
SHR  dest, quantity
SUB  dest, arg
JC   etykieta
JNC  etykieta
JMP  etykieta
```

```
if( a < 0x05 )
    y = 0;
else
    y = 1;
```

Wersja B:

```
MOV  dest, source
SHL  dest, quantity
SHR  dest, quantity
SUB  dest, arg
JC   etykieta
JNC  etykieta
STC
CLC
```

Wersja C:

```
MOV  dest, source
SHL  dest, quantity
SHR  dest, quantity
CMP  dest, arg
JC   etykieta
JNC  etykieta
STC
CLC
```

Wersja D:

```
MOV  dest, source
SHL  dest, quantity
SHR  dest, quantity
ADD  dest, arg
JC   etykieta
JNC  etykieta
JMP  etykieta
```

Wersja E:

```
if( a <= 0x05 )
    y = 0;
else
    y = 1;
```

Wersja F:

```
if( a > 0x05 )
    y = 0;
else
    y = 1;
```

Zadanie 12: Napisz kod w asemblerze odpowiadający poniższemu fragmentowi kodu w C.

Zmienna ***i*** oraz ***y*** 32-bitowe.
Można wykorzystywać tylko instrukcje z listy poniżej.

Wersja A:

```
MOV  dest, source
SUB  dest, arg
ADD  dest, arg
JC       etykieta
JNC    etykieta
JMP    etykieta
XOR  dest, arg
```

```
for( int i=0; i<5; i++ ) y += i;
```

Wersja B:

```
MOV  dest, source
CMP  dest, arg
ADD  dest, arg
JC       etykieta
JNC    etykieta
JMP    etykieta
XOR  dest, arg
```

Wersja C:

```
for( int i=5; i>=1; i-- ) y -= i;
```

Zadanie 13: Napisz kod w C (bez użycia pól bitowych) oraz w asemblerze odpowiadający poniższemu fragmentowi kodu w C.

Można wykorzystywać tylko instrukcje z listy poniżej.

```
MOV  dest, source
SHL  dest, quantity
SHR  dest, quantity
ADD  dest, arg
SUB  dest, arg
JZ     etykieta
JNC   etykieta
AND  dest, arg
OR  dest, arg
XOR  dest, arg
```

```
typedef union {
     unsigned char  BYTE;
     struct{
          unsigned char a:2;
          unsigned char b:3;
          unsigned char c:3;
     }bits;
}un_X;

un_X x;
```

```
if( x.bits.c == 1 ) x.bits.a = x.bits.b;
```

Zadanie 14: Napisz kod w C (bez użycia pól bitowych) oraz w asemblerze odpowiadający poniższemu fragmentowi kodu w C.

Można wykorzystywać tylko instrukcje z listy poniżej.

```
MOV  dest, source
SHL   dest, quantity
SHR   dest, quantity
ADD  dest, arg
SUB  dest, arg
JZ       etykieta
JNC    etykieta
AND  dest, arg
OR     dest, arg
XOR  dest, arg
```

```
typedef union {
     unsigned char  BYTE;
     struct{
          unsigned char a:1;
          unsigned char b:3;
          unsigned char c:4;
     }bits;
}un_X;

un_X x;
```

```
x.bits.c = x.bits.a * x.bits.b;
```

Zadanie 15: Napisz w języku C kod odpowiadający poniższemu fragmentowi kodu w C (bez użycia adresowania indeksowego, tylko na bazie wskaźników). Następnie napisz kod w asemblerze dla obu fragmentów kodu w C.

```
unsigned char tab[5];
for( int i=0; i!=5; i++ ) tab[i] = i;
```

```
MOV  dest, source
SHL  dest, quantity
SHR  dest, quantity
ADD  dest, arg
SUB  dest, arg
CMP  dest, arg
JZ   etykieta
JNZ  etykieta
JC   etykieta
JNC  etykieta
JMP  etykieta
AND  dest, arg
OR   dest, arg
XOR  dest, arg
TEST dest, arg
```

Adresowanie pośrednie oznaczamy []

```
lea edi, tab
```

Wersja B:

```
unsigned char tab[5];
for( int i=0; i<5; i++ ) tab[i] = i;
```

Zadanie 16: Napisz kod pokazujący wartość struktury bajt po bajcie oraz jej rozmiar. Następnie napisz w języku C kod odpowiadający poniższemu fragmentowi kodu w C (bez użycia odwołania do pól struktury). Następnie napisz kod w asemblerze dla obu fragmentów kodu w C.

Adresowanie pośrednie oznaczamy []

```
MOV  dest, source
SHL  dest, quantity
SHR  dest, quantity
ADD  dest, arg
SUB  dest, arg
CMP  dest, arg
JZ   etykieta
JNZ  etykieta
JC   etykieta
JNC  etykieta
JMP  etykieta
AND  dest, arg
OR   dest, arg
XOR  dest, arg
TEST dest, arg
```

```
#pragma pack(1)
typedef struct{
  unsigned char a;
  int b;
  long c;
}MTestStruct;
```

```
MTestStruct testStruct;
testStruct.a = 1;
testStruct.b = 2;
testStruct.c = 3;
```

Wersja B: bez #pragma pack(1) – co się zmieniło

```
typedef struct{
  unsigned char a;
  int b;
  long c;
}MTestStruct;
```

Zadanie 17: Napisz kod pokazujący wartość struktury bajt po bajcie oraz jej rozmiar. Następnie napisz w języku C kod odpowiadający poniższemu fragmentowi kodu w C (bez użycia odwołania do pól struktury). Następnie napisz kod w asemblerze dla obu fragmentów kodu w C.

Adresowanie pośrednie oznaczamy []

```
MOV  dest, source
SHL  dest, quantity
SHR  dest, quantity
ADD  dest, arg
SUB  dest, arg
CMP  dest, arg
JZ     etykieta
JNZ    etykieta
JC     etykieta
JNC    etykieta
JMP    etykieta
AND  dest, arg
OR   dest, arg
XOR  dest, arg
TEST dest, arg
```

```
class Base{
public:
    unsigned char a;
};
class MTestStruct : public Base{
public:
   int b;
   long c;
};
```

```
MTestStruct testStruct;
testStruct.a = 1;
testStruct.b = 2;
testStruct.c = 3;
```

Wersja B: private – czy kod asemblerowy bezpośredni odczyta wartość pola private ?

```
class Base{
private:
    unsigned char a;
};
```

Zadanie 18: Napisz kod pokazujący wartość struktury bajt po bajcie oraz jej rozmiar. Następnie napisz w języku C kod odpowiadający poniższemu fragmentowi kodu w C (bez użycia odwołania do pól struktury). Następnie napisz kod w asemblerze dla obu fragmentów kodu w C. Porównaj z wynikami z zadania 16.

Adresowanie pośrednie oznaczamy []

```
MOV  dest, source
SHL  dest, quantity
SHR  dest, quantity
ADD  dest, arg
SUB  dest, arg
CMP  dest, arg
JZ   etykieta
JNZ  etykieta
JC   etykieta
JNC  etykieta
JMP  etykieta
AND  dest, arg
OR   dest, arg
XOR  dest, arg
TEST dest, arg
```

```
class Base{
public:
    unsigned char a;
    virtual void aaa() {};

};
class MTestStruct : public Base{
public:
   int b;
   long c;
   virtual void aaa() {};

};
```

```
MTestStruct testStruct;
testStruct.a = 1;
testStruct.b = 2;
testStruct.c = 3;
```

Zadanie 19: Napisz kod w asemblerze odpowiadający poniższemu fragmentowi kodu w C (w asemblerze tylko wywołanie funkcji myABS, instrukcja w czerwonej ramce). Jaki jest domyślny format wywołania funkcji: stdcall czy cdecl (sprawdź)?

Wersja B: ***stdcall***

```
unsigned char _stdcall myABS(char a) {
	if (a > 0) return a;
	else return -a;
}

int main()
{
	char a = -15;
	unsigned char y;

	y = myABS(a);

	return y;
```

Wersja B: ***cdecl***

```
unsigned char _cdecl myABS(char a) {
	if (a > 0) return a;
	else return -a;
}

int main()
{
	char a = -15;
	unsigned char y;

	y = myABS(a);

	return y;
```