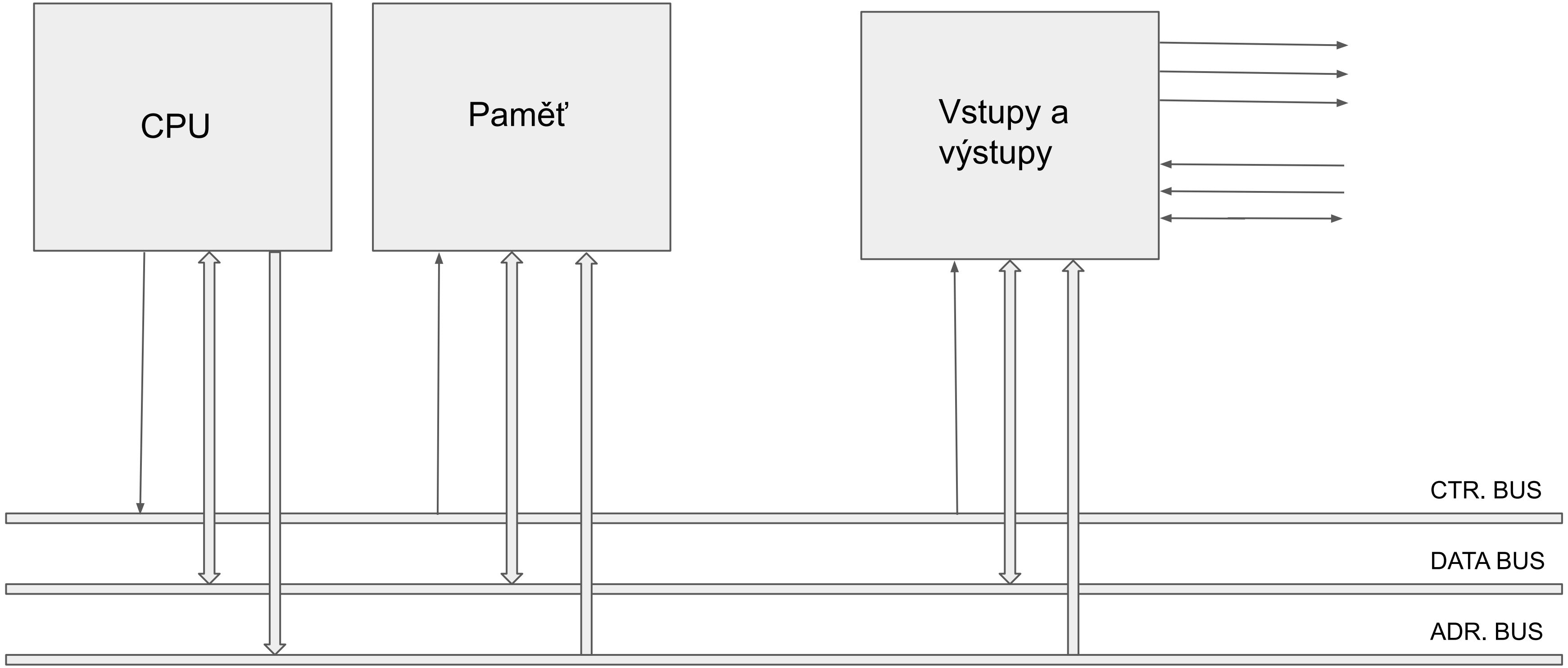
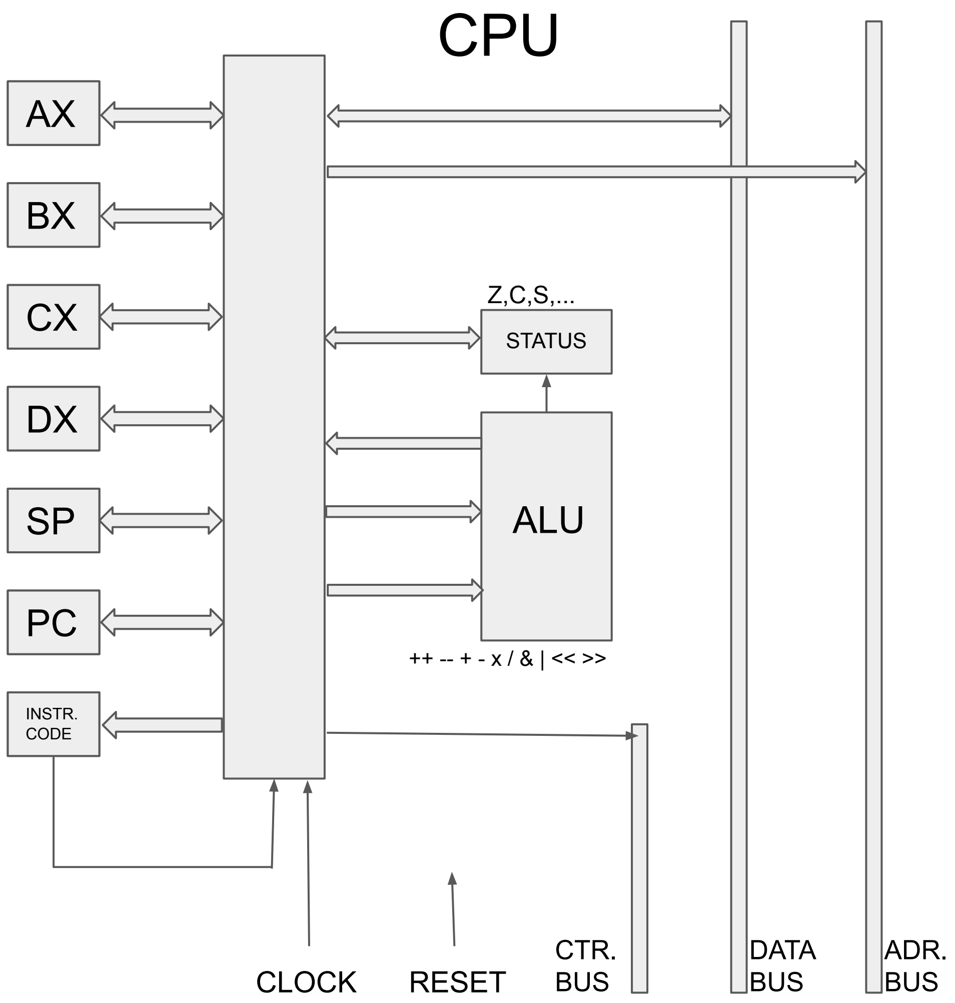
Schéma počítače (upravený model von Neumanna)





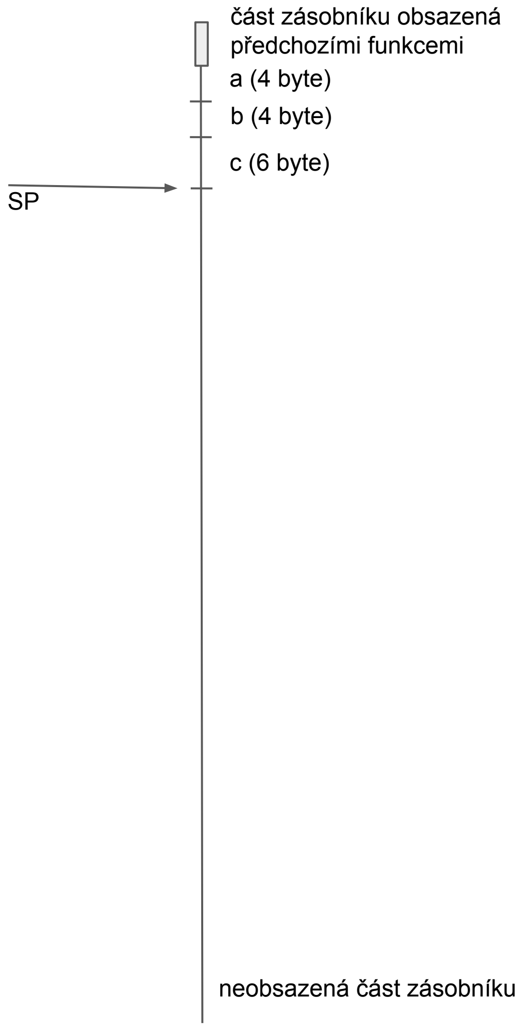
Instrukce (minimální sada)
op2 registr, konstanta
op2 registr, registr
op2 registr, [adresa]
op2 registr, [registr + konstanta]
op2 jsou: LOAD, STORE, ADD, SUB, CMP,...

op1 registr
op1 [adresa]
op1 [registr + konstanta]
op1 jsou: INC, DEC, CLEAR, ...

PUSH registr
POP registr
skok konstanta
skok registr
operace skoku jsou: JMP, CALL, JC, JNC, JZ, …
RETURN

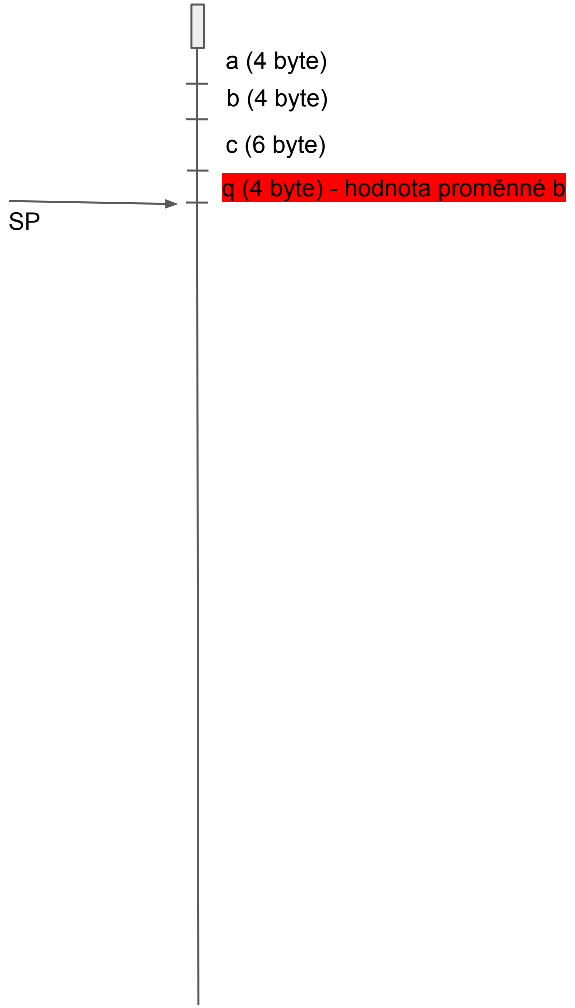
```
int main() {
int a;
int b;
char c[5 + 1];

…
a = fce(5 * 3, b);
…

}

int fce(int p, int q) {
int r;
char s[10+1];

r = p + 10;
s[5] = p;
s[q + 1] = '\n';
return r;
}
```

```
main:
…
LOAD AX,[SP + 6]
PUSH AX
LOAD AX, 15
PUSH AX
CALL fce
ADD SP,8
STORE AX,[SP + 10]
…

fce:
ADD SP,-15
LOAD AX,[SP + 15 + 4 + 4]
ADD AX,10
STORE AX,[SP + 11]
LOAD AX,[SP + 15 + 4 + 4]
STORE AL,[SP + 5]
LOAD BX,[SP + 15 + 4 + 0]
INC BX
ADD BX,SP
MOV AL,10
STORE AL,[BX]
LOAD AX,[SP + 11]
ADD SP,15
RETURN
```


část zásobníku obsazená předchozími funkcemi
a (4 byte)
b (4 byte)
c (6 byte)
SP
neobsazená část zásobníku

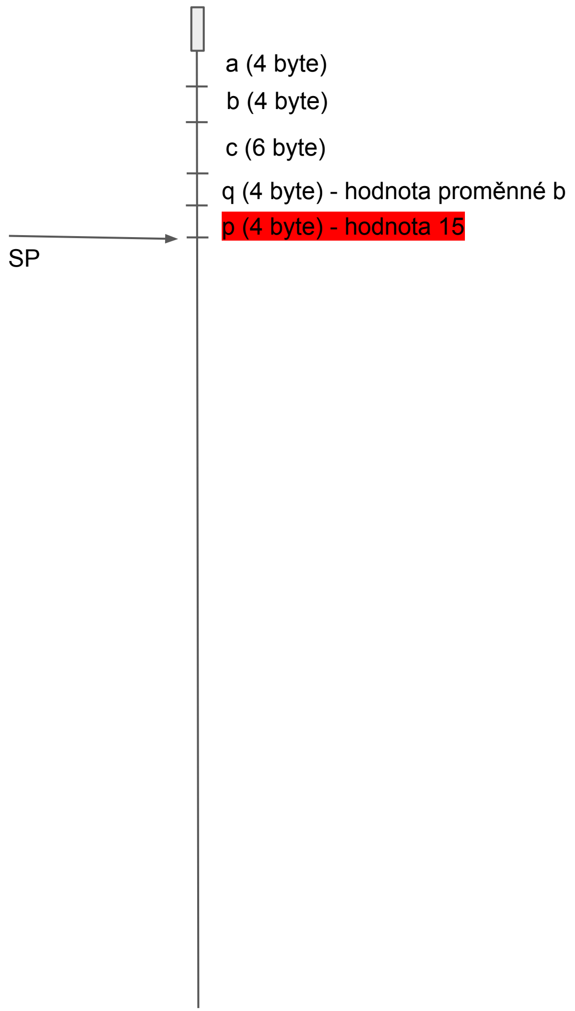
```
int main() {
int a;
int b;
char c[5 + 1];

…
a = fce(5 * 3, b);
…

}

int fce(int p, int q) {
int r;
char s[10+1];

r = p + 10;
s[5] = p;
s[q + 1] = '\n';
return r;
}
```

```
main:
…
LOAD AX,[SP + 6]
PUSH AX
LOAD AX, 15
PUSH AX
CALL fce
ADD SP,8
STORE AX,[SP + 10]
…

fce:
ADD SP,-15
LOAD AX,[SP + 15 + 4 + 4]
ADD AX,10
STORE AX,[SP + 11]
LOAD AX,[SP + 15 + 4 + 4]
STORE AL,[SP + 5]
LOAD BX,[SP + 15 + 4 + 0]
INC BX
ADD BX,SP
MOV AL,10
STORE AL,[BX]
LOAD AX,[SP + 11]
ADD SP,15
RETURN
```



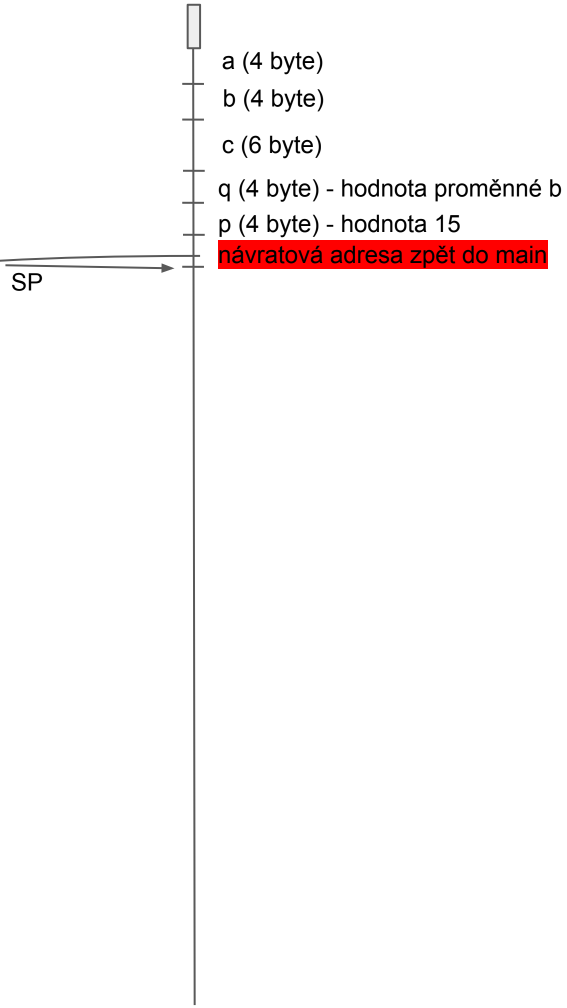
```
int main() {
int a;
int b;
char c[5 + 1];

…
a = fce(5 * 3, b);
…

}

int fce(int p, int q) {
int r;
char s[10+1];

r = p + 10;
s[5] = p;
s[q + 1] = '\n';
return r;
}
```

```
main:
…
LOAD AX,[SP + 6]
PUSH AX
LOAD AX, 15
PUSH AX
CALL fce
ADD SP,8
STORE AX,[SP + 10]
…

fce:
ADD SP,-15
LOAD AX,[SP + 15 + 4 + 4]
ADD AX,10
STORE AX,[SP + 11]
LOAD AX,[SP + 15 + 4 + 4]
STORE AL,[SP + 5]
LOAD BX,[SP + 15 + 4 + 0]
INC BX
ADD BX,SP
MOV AL,10
STORE AL,[BX]
LOAD AX,[SP + 11]
ADD SP,15
RETURN
```



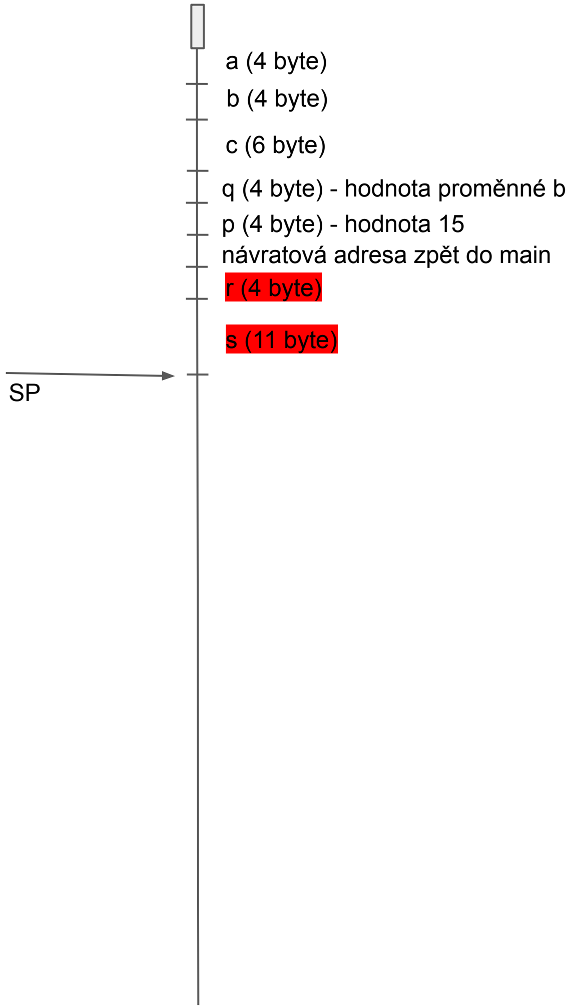
```
int main() {
int a;
int b;
char c[5 + 1];

…
a = fce(5 * 3, b);
…

}

int fce(int p, int q) {
int r;
char s[10+1];

r = p + 10;
s[5] = p;
s[q + 1] = '\n';
return r;
}
```

```
main:
…
LOAD AX,[SP + 6]
PUSH AX
LOAD AX, 15
PUSH AX
CALL fce
ADD SP,8
STORE AX,[SP + 10]
…

fce:
ADD SP,-15
LOAD AX,[SP + 15 + 4 + 4]
ADD AX,10
STORE AX,[SP + 11]
LOAD AX,[SP + 15 + 4 + 4]
STORE AL,[SP + 5]
LOAD BX,[SP + 15 + 4 + 0]
INC BX
ADD BX,SP
MOV AL,10
STORE AL,[BX]
LOAD AX,[SP + 11]
ADD SP,15
RETURN
```


a (4 byte)
b (4 byte)
c (6 byte)
q (4 byte) - hodnota proměnné b
p (4 byte) - hodnota 15
návratová adresa zpět do main
SP

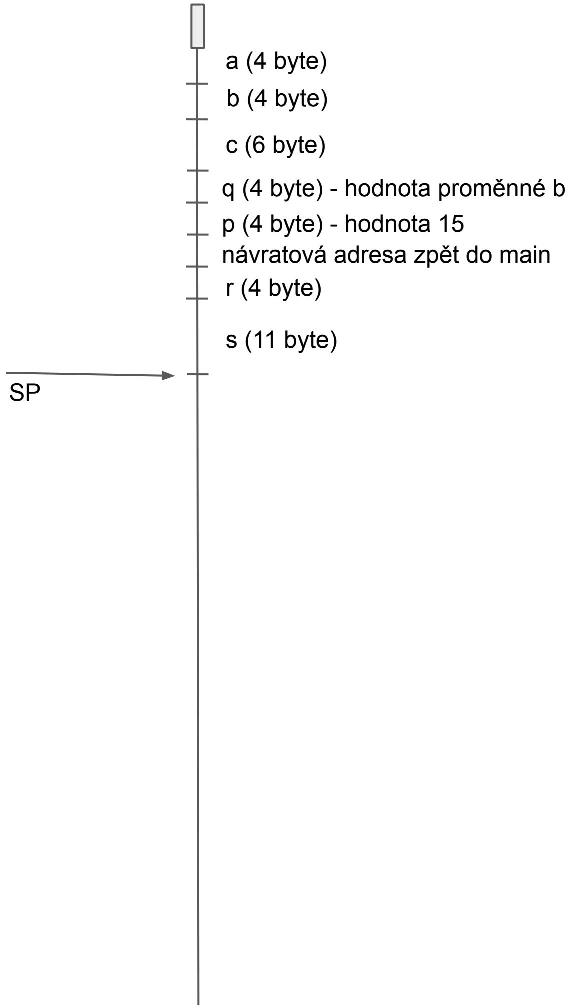
```
int main() {
int a;
int b;
char c[5 + 1];

…
a = fce(5 * 3, b);
…

}

int fce(int p, int q) {
int r;
char s[10+1];

r = p + 10;
s[5] = p;
s[q + 1] = '\n';
return r;
}
```

```
main:
…
LOAD AX,[SP + 6]
PUSH AX
LOAD AX, 15
PUSH AX
CALL fce
ADD SP,8
STORE AX,[SP + 10]
…

fce:
ADD SP,-15
LOAD AX,[SP + 15 + 4 + 4]
ADD AX,10
STORE AX,[SP + 11]
LOAD AX,[SP + 15 + 4 + 4]
STORE AL,[SP + 5]
LOAD BX,[SP + 15 + 4 + 0]
INC BX
ADD BX,SP
MOV AL,10
STORE AL,[BX]
LOAD AX,[SP + 11]
ADD SP,15
RETURN
```



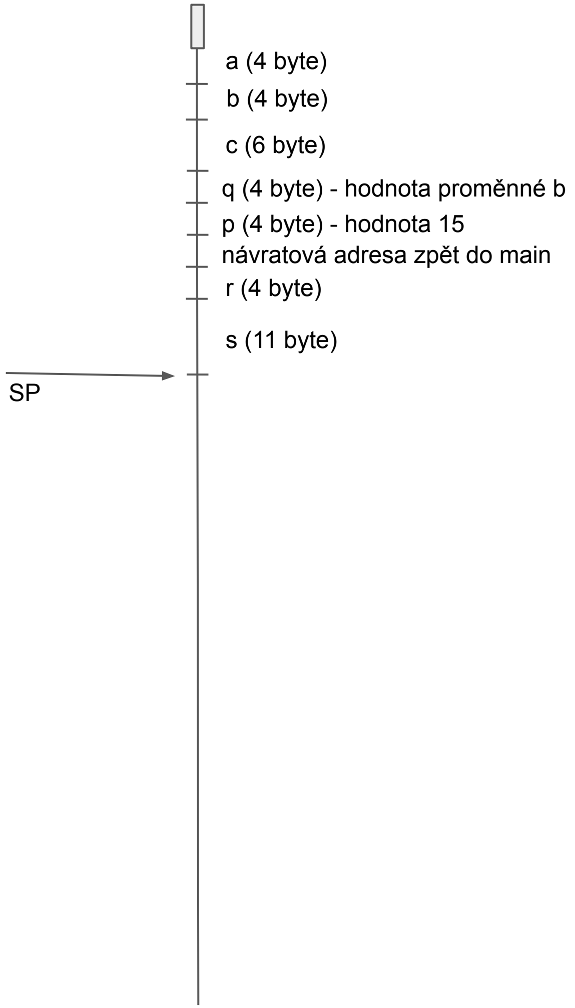
```
int main() {
int a;
int b;
char c[5 + 1];

…
a = fce(5 * 3, b);
…

}

int fce(int p, int q) {
int r;
char s[10+1];

r = p + 10;
s[5] = p;
s[q + 1] = '\n';
return r;
}
```

```
main:
…
LOAD AX,[SP + 6]
PUSH AX
LOAD AX, 15
PUSH AX
CALL fce
ADD SP,8
STORE AX,[SP + 10]
…

fce:
ADD SP,-15
LOAD AX,[SP + 15 + 4 + 4]
ADD AX,10
STORE AX,[SP + 11]
LOAD AX,[SP + 15 + 4 + 4]
STORE AL,[SP + 5]
LOAD BX,[SP + 15 + 4 + 0]
INC BX
ADD BX,SP
MOV AL,10
STORE AL,[BX]
LOAD AX,[SP + 11]
ADD SP,15
RETURN
```



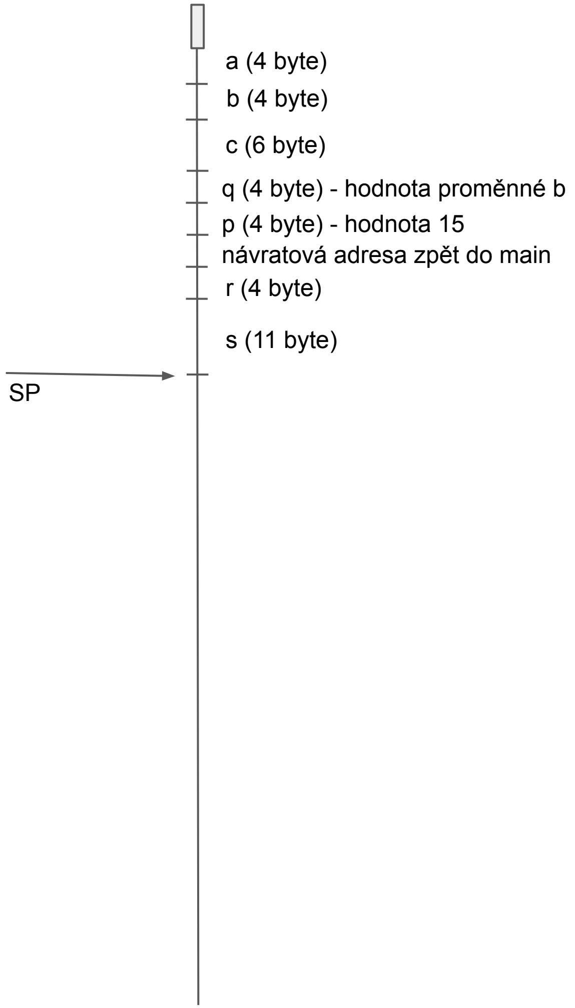
```
int main() {
int a;
int b;
char c[5 + 1];

…
a = fce(5 * 3, b);
…

}

int fce(int p, int q) {
int r;
char s[10+1];

r = p + 10;
s[5] = p;
s[q + 1] = '\n';
return r;
}
```

```
main:
…
LOAD AX,[SP + 6]
PUSH AX
LOAD AX, 15
PUSH AX
CALL fce
ADD SP,8
STORE AX,[SP + 10]
…

fce:
ADD SP,-15
LOAD AX,[SP + 15 + 4 + 4]
ADD AX,10
STORE AX,[SP + 11]
LOAD AX,[SP + 15 + 4 + 4]
STORE AL,[SP + 5]
LOAD BX,[SP + 15 + 4 + 0]
INC BX
ADD BX,SP
MOV AL,10
STORE AL,[BX]
LOAD AX,[SP + 11]
ADD SP,15
RETURN
```



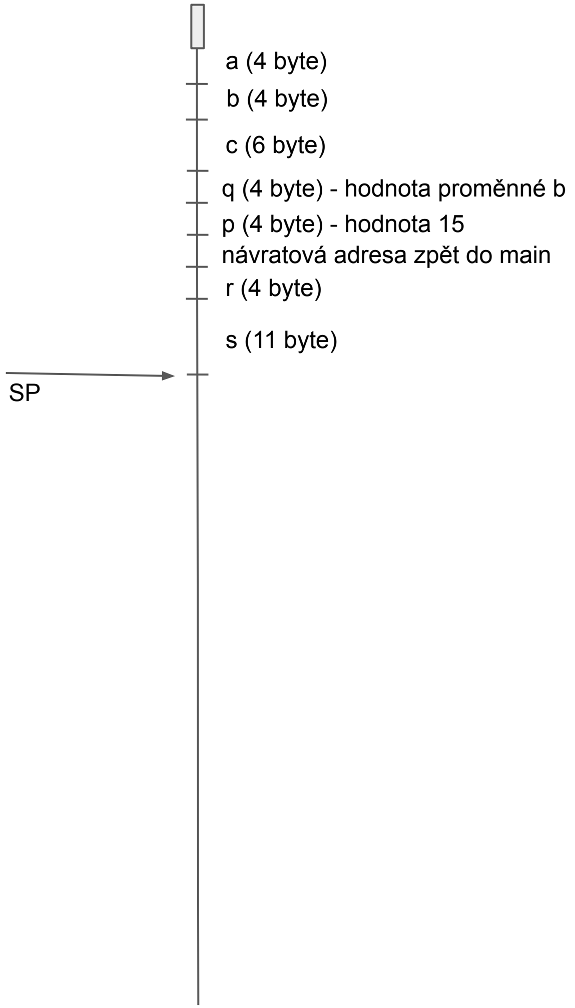
```
int main() {
int a;
int b;
char c[5 + 1];

…
a = fce(5 * 3, b);
…

}

int fce(int p, int q) {
int r;
char s[10+1];

r = p + 10;
s[5] = p;
s[q + 1] = '\n';
return r;
}
```

```
main:
…
LOAD AX,[SP + 6]
PUSH AX
LOAD AX, 15
PUSH AX
CALL fce
ADD SP,8
STORE AX,[SP + 10]
…

fce:
ADD SP,-15
LOAD AX,[SP + 15 + 4 + 4]
ADD AX,10
STORE AX,[SP + 11]
LOAD AX,[SP + 15 + 4 + 4]
STORE AL,[SP + 5]
LOAD BX,[SP + 15 + 4 + 0]
INC BX
ADD BX,SP
MOV AL,10
STORE AL,[BX]
LOAD AX,[SP + 11]
ADD SP,15
RETURN
```



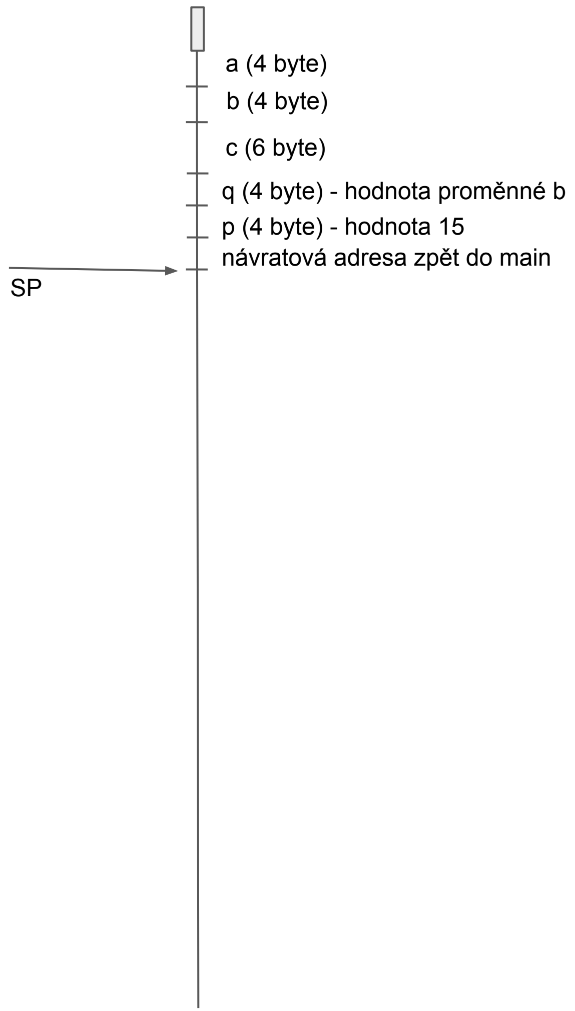
```
int main() {
int a;
int b;
char c[5 + 1];

…
a = fce(5 * 3, b);
…

}

int fce(int p, int q) {
int r;
char s[10+1];

r = p + 10;
s[5] = p;
s[q + 1] = '\n';
return r;
}
```

```
main:
…
LOAD AX,[SP + 6]
PUSH AX
LOAD AX, 15
PUSH AX
CALL fce
ADD SP,8
STORE AX,[SP + 10]
…

fce:
ADD SP,-15
LOAD AX,[SP + 15 + 4 + 4]
ADD AX,10
STORE AX,[SP + 11]
LOAD AX,[SP + 15 + 4 + 4]
STORE AL,[SP + 5]
LOAD BX,[SP + 15 + 4 + 0]
INC BX
ADD BX,SP
MOV AL,10
STORE AL,[BX]
LOAD AX,[SP + 11]
ADD SP,15
RETURN
```



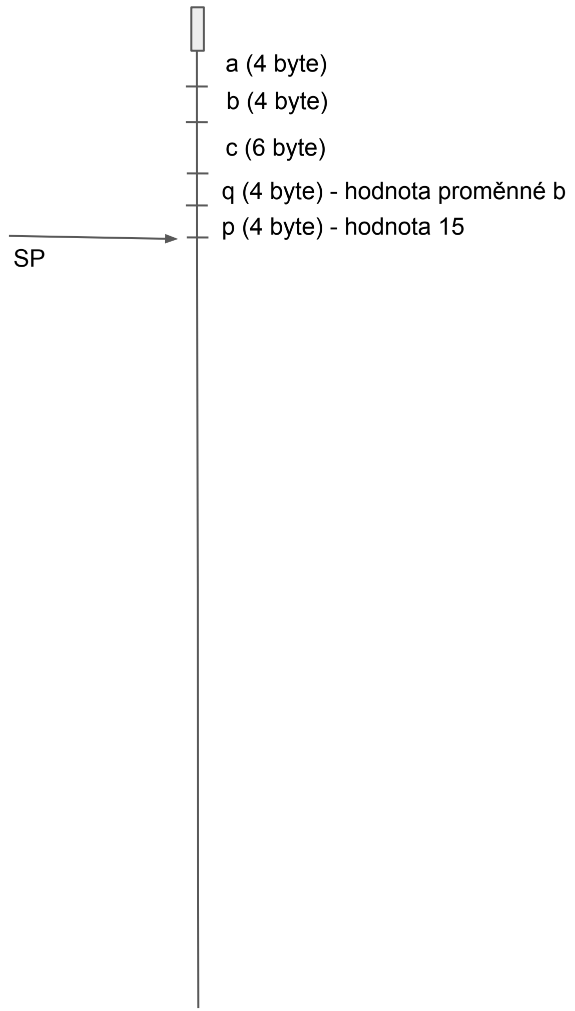
```
int main() {
int a;
int b;
char c[5 + 1];

…
a = fce(5 * 3, b);
…

}

int fce(int p, int q) {
int r;
char s[10+1];

r = p + 10;
s[5] = p;
s[q + 1] = '\n';
return r;
}
```

```
main:
…
LOAD AX,[SP + 6]
PUSH AX
LOAD AX, 15
PUSH AX
CALL fce
ADD SP,8
STORE AX,[SP + 10]
…

fce:
ADD SP,-15
LOAD AX,[SP + 15 + 4 + 4]
ADD AX,10
STORE AX,[SP + 11]
LOAD AX,[SP + 15 + 4 + 4]
STORE AL,[SP + 5]
LOAD BX,[SP + 15 + 4 + 0]
INC BX
ADD BX,SP
MOV AL,10
STORE AL,[BX]
LOAD AX,[SP + 11]
ADD SP,15
RETURN
```


a (4 byte)
b (4 byte)
c (6 byte)
q (4 byte) - hodnota proměnné b
p (4 byte) - hodnota 15
návratová adresa zpět do main
SP

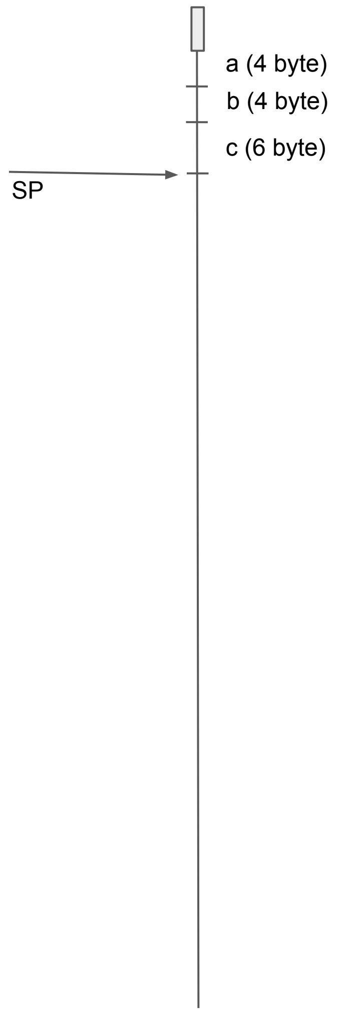
```
int main() {
int a;
int b;
char c[5 + 1];

…
a = fce(5 * 3, b);
…

}

int fce(int p, int q) {
int r;
char s[10+1];

r = p + 10;
s[5] = p;
s[q + 1] = '\n';
return r;
}
```

```
main:
…
LOAD AX,[SP + 6]
PUSH AX
LOAD AX, 15
PUSH AX
CALL fce
ADD SP,8
STORE AX,[SP + 10]
…

fce:
ADD SP,-15
LOAD AX,[SP + 15 + 4 + 4]
ADD AX,10
STORE AX,[SP + 11]
LOAD AX,[SP + 15 + 4 + 4]
STORE AL,[SP + 5]
LOAD BX,[SP + 15 + 4 + 0]
INC BX
ADD BX,SP
MOV AL,10
STORE AL,[BX]
LOAD AX,[SP + 11]
ADD SP,15
RETURN
```



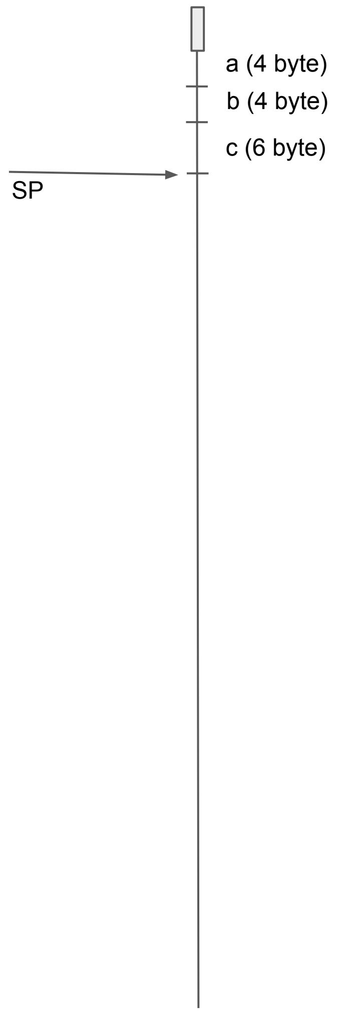
```
int main() {
int a;
int b;
char c[5 + 1];

…
a = fce(5 * 3, b);
…

}

int fce(int p, int q) {
int r;
char s[10+1];

r = p + 10;
s[5] = p;
s[q + 1] = '\n';
return r;
}
```

```
main:
…
LOAD AX,[SP + 6]
PUSH AX
LOAD AX, 15
PUSH AX
CALL fce
ADD SP,8
STORE AX,[SP + 10]
…

fce:
ADD SP,-15
LOAD AX,[SP + 15 + 4 + 4]
ADD AX,10
STORE AX,[SP + 11]
LOAD AX,[SP + 15 + 4 + 4]
STORE AL,[SP + 5]
LOAD BX,[SP + 15 + 4 + 0]
INC BX
ADD BX,SP
MOV AL,10
STORE AL,[BX]
LOAD AX,[SP + 11]
ADD SP,15
RETURN
```



```
int main() {
int a;
int b;
char c[5 + 1];

…
a = fce(5 * 3, b);
…

}

int fce(int p, int q) {
int r;
char s[10+1];

r = p + 10;
s[5] = p;
s[q + 1] = '\n';
return r;
}
```

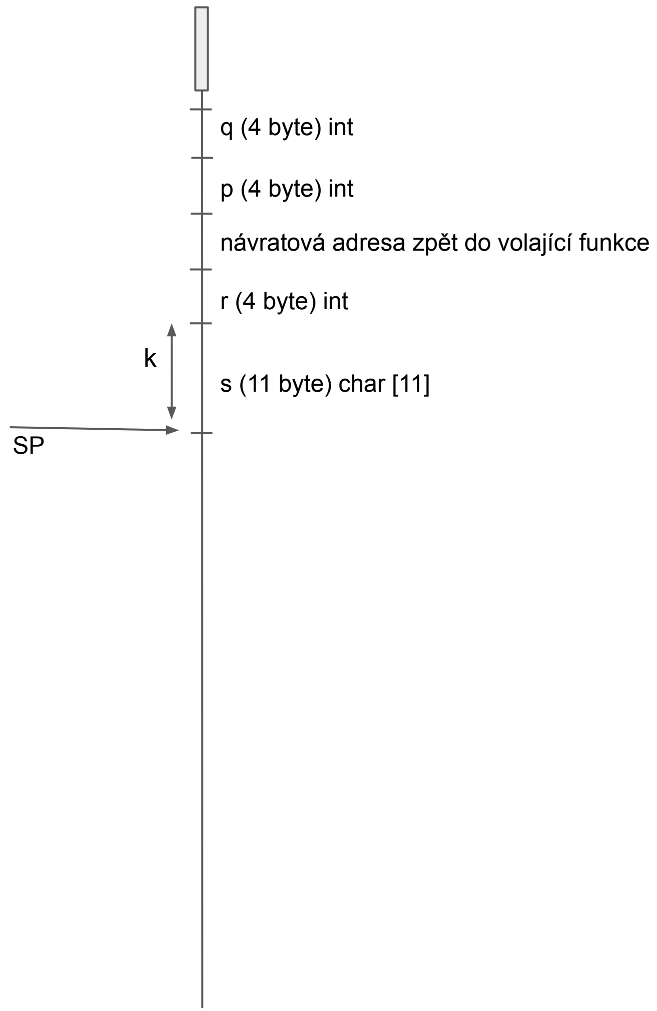
```
main:
…
LOAD AX,[SP + 6]
PUSH AX
LOAD AX, 15
PUSH AX
CALL fce
ADD SP,8
STORE AX,[SP + 10]
…

fce:
ADD SP,-15
LOAD AX,[SP + 15 + 4 + 4]
ADD AX,10
STORE AX,[SP + 11]
LOAD AX,[SP + 15 + 4 + 4]
STORE AL,[SP + 5]
LOAD BX,[SP + 15 + 4 + 0]
INC BX
ADD BX,SP
MOV AL,10
STORE AL,[BX]
LOAD AX,[SP + 11]
ADD SP,15
RETURN
```



Zpřístupnění dat

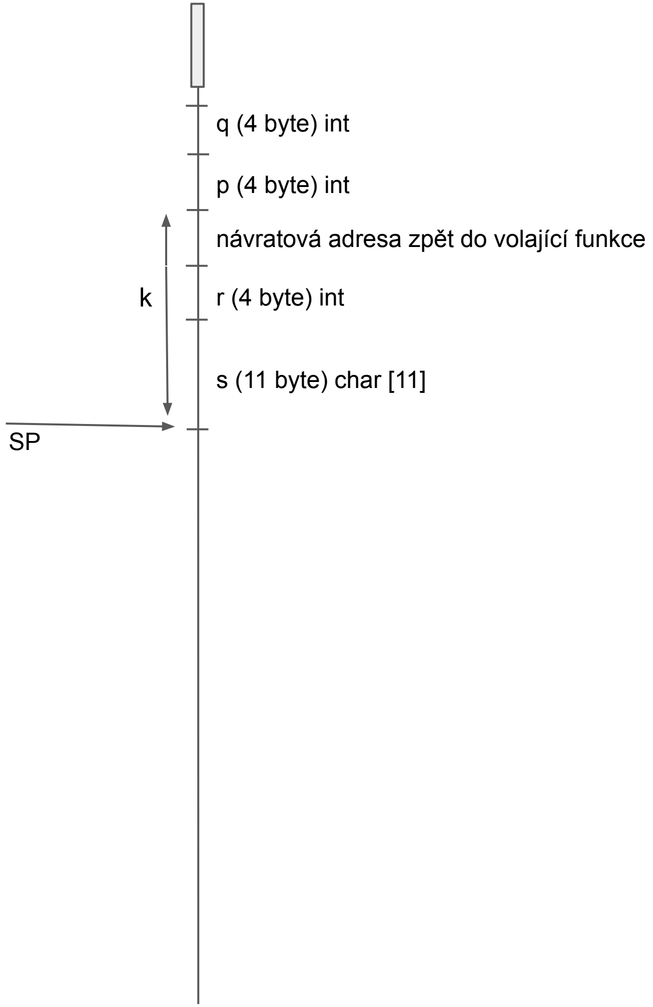
Lokální proměnná

```
int r;

LOAD AX,[SP + k]
```



Parametr

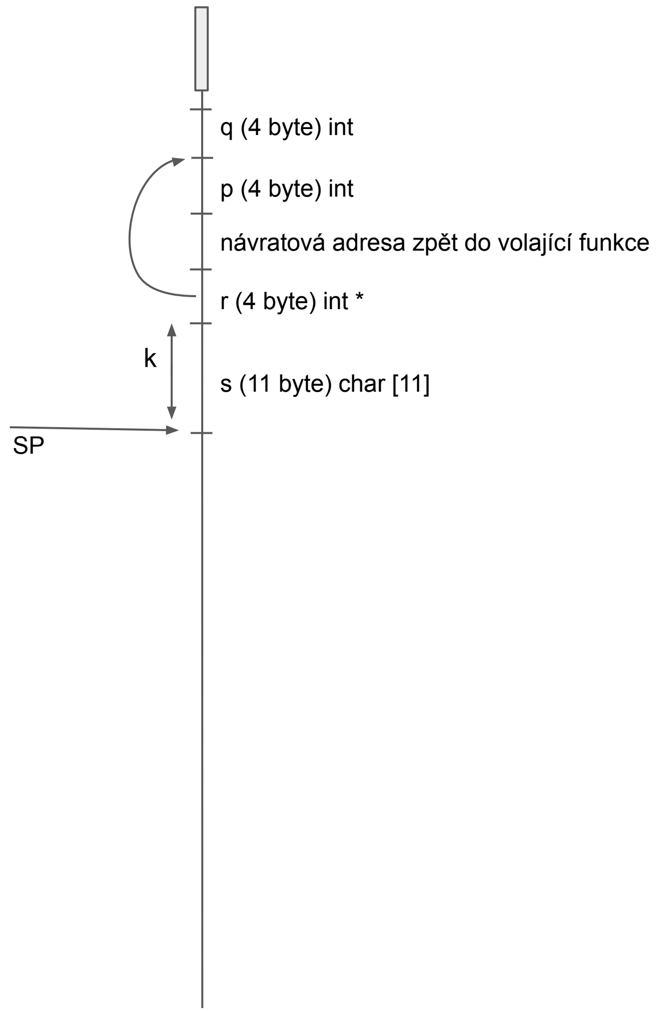
```
void fce(int p, int q)

LOAD AX,[SP + k]
```



Reference přes pointer

```
int * r = &q;
```
`*r`

```
LOAD BX,[SP + k]
LOAD AX,[BX]
```

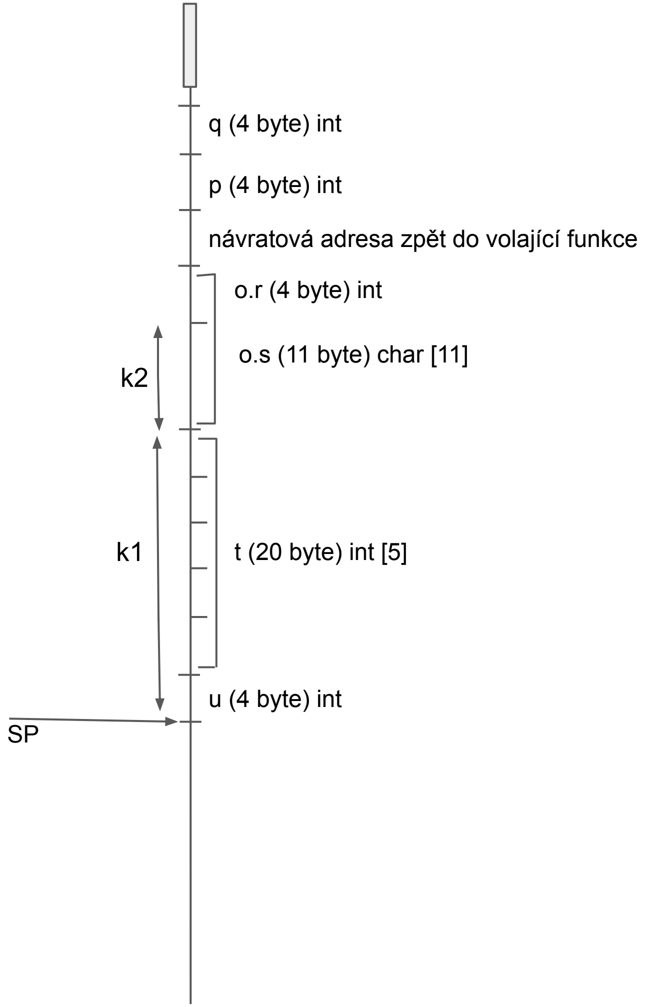


Atribut třídy, prvek struktury (jako lokální proměnná)

```
struct x {
  int r;
  char s[10+1];
} o;
```

o.r

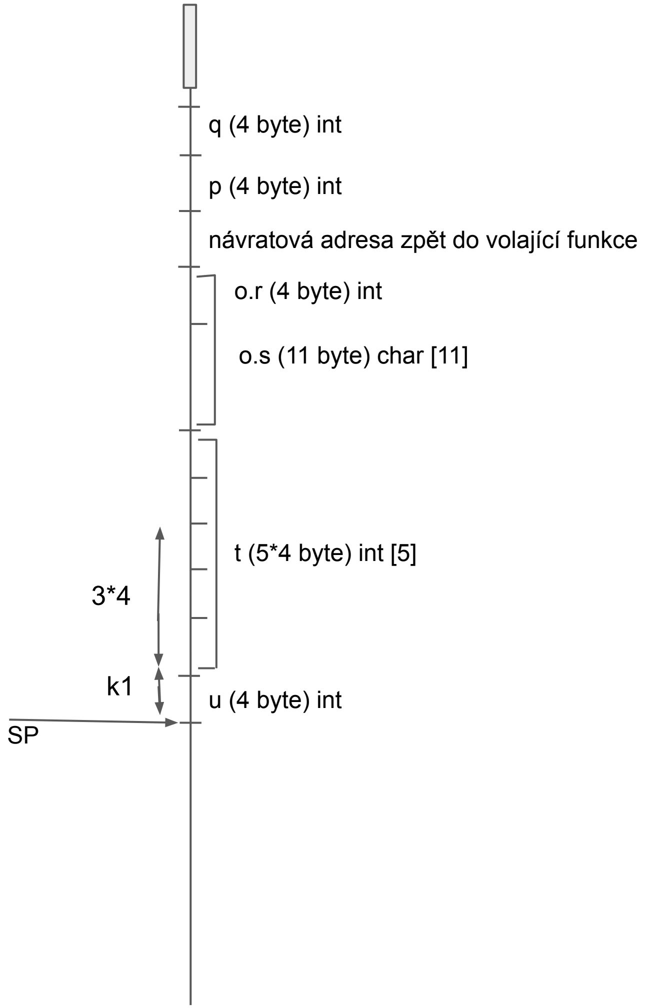
```
LOAD AX,[SP + (k1 + k2)]
```



Prvek pole s konstantním indexem (jako lokální proměnná)

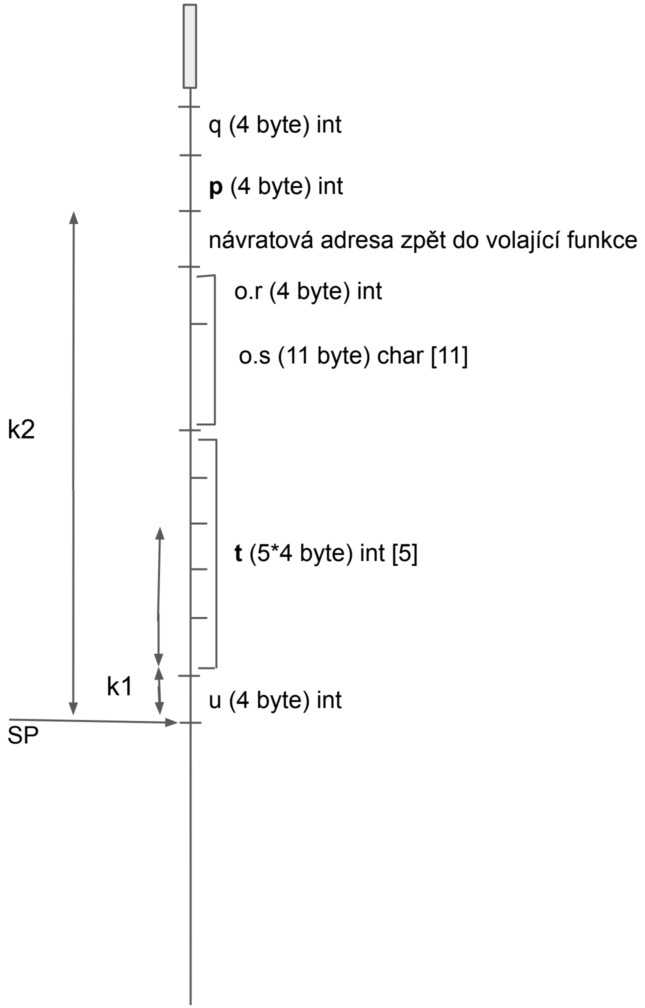
`t[3]`

```
LOAD AX,[SP + (k1 + 3*4)]
```



Prvek pole s nekonstantním indexem (jako lokální proměnná)

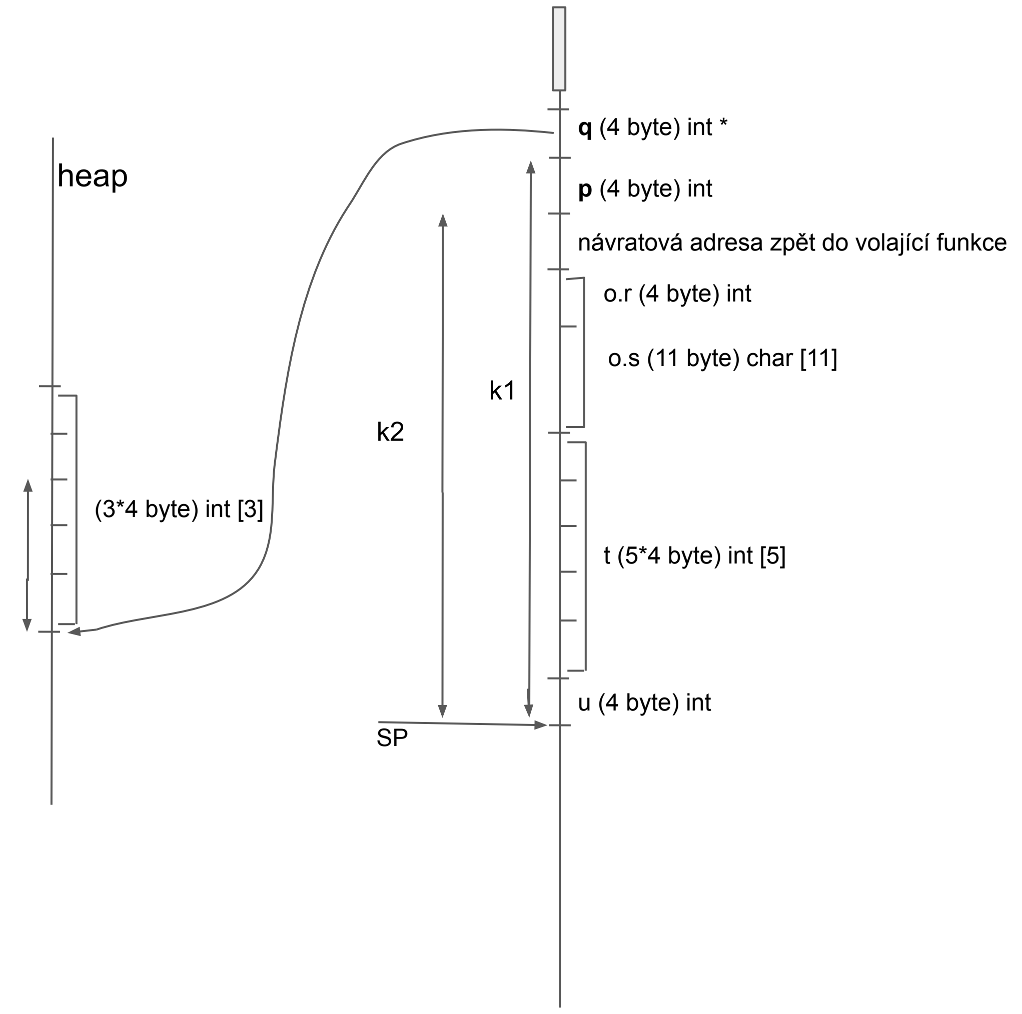
```
int p = 3
t[p]

LOAD BX,[SP + k2]
MUL BX,k3           k3 = sizeof(int)
ADD BX,k1
LOAD AX,[SP + BX]

alternativně poslední instrukce:
ADD BX, SP
LOAD AX,[BX]
```



Prvek pole s nekonstantním indexem (na haldě)

```
int * q
int p = 3
q[p]
```

```
LOAD BX,[SP + k2]
MUL BX,k3            k3 = sizeof(int)
ADD BX,[SP + k1]
LOAD AX,[BX]
```

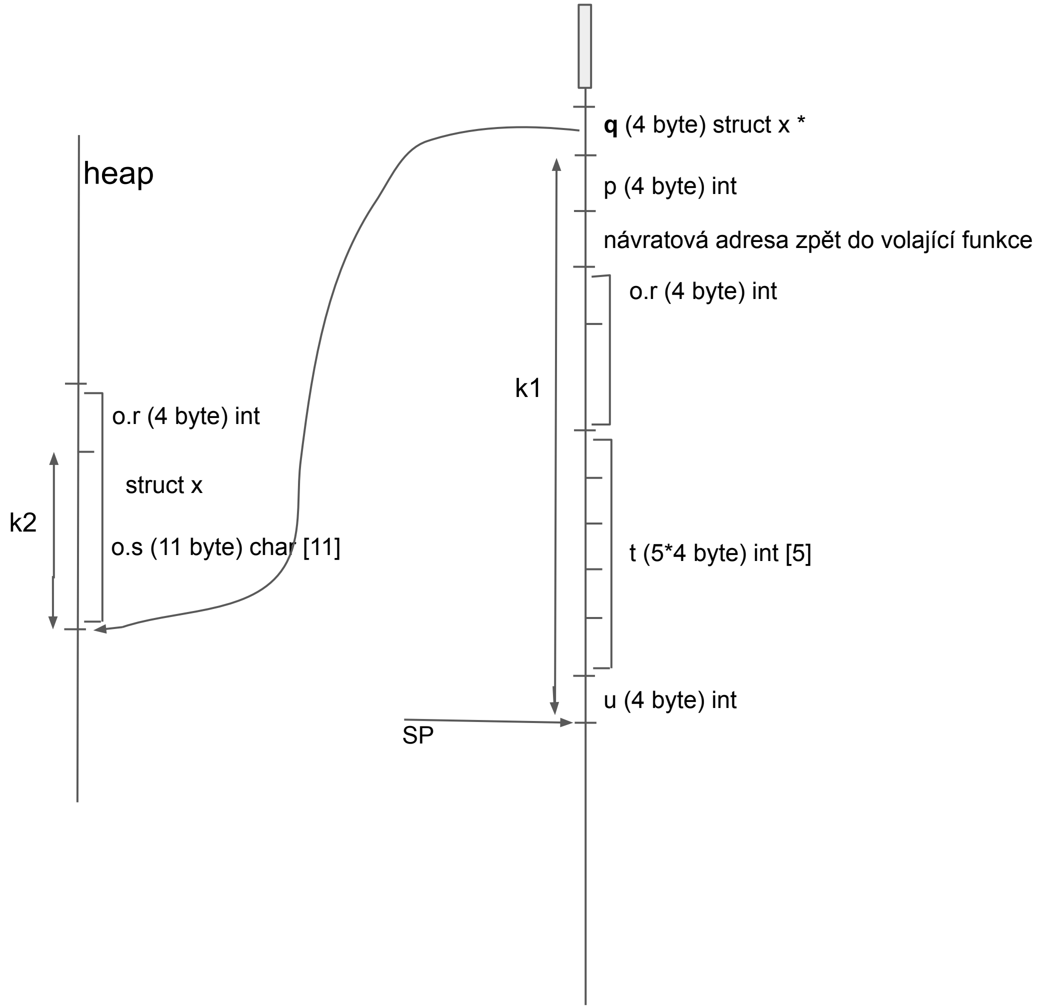


Atribut třídy, prvek struktury (na haldě)

```
struct x {
  int r;
  char s[10+1];
} o;
```

q.r

```
LOAD BX,[SP + k1]
ADD BX,k2
LOAD AX,[BX]
```


heap
o.r (4 byte) int
struct x
o.s (11 byte) char [11]
k2
q (4 byte) struct x *
p (4 byte) int
návratová adresa zpět do volající funkce
o.r (4 byte) int
k1
t (5*4 byte) int [5]
u (4 byte) int
SP