```
int m1[M][N];
int m2[N][M];

int sum_element(int i, int j) {
   return m1[i][j] + m2[j][i];
}
```

```
; i – [ebp + 8]
; j – [ebp + 12]
;
;
; …
 mov  ecx, dword [ebp + 8]       ; 1
 mov  edx, dword [ebp + 12]      ; 2
 lea  eax, [8 * ecx]             ; 3
 sub  eax, ecx                   ; 4
 add  eax, edx                   ; 5
 lea  edx, [edx + 4 * edx]       ; 6
 add  edx, ecx                   ; 7
 mov  eax, dword [m1 + 4 * eax] ; 8
 add  eax, dword [m2 + 4 * edx] ; 9
 ; …
```

```
enum TargetPosition {
    TARGET_AT_BEGINNING,
    TARGET_AT_MIDDLE,
    TARGET_AT_END
 };

 switch (targetPosition){

 case TARGET_AT_BEGINNING:
    offsetInWindow = 0;
    break;
 case TARGET_AT_MIDDLE:
    offsetInWindow = MIN (newSize - size, newSize / 2);
    break;
 case TARGET_AT_END:
    offsetInWindow = newSize - size;
    break;
 default:
    _error_code = IllegalData;
    _error_msg  = tr("requested target position"
                     " is not a TargetPosition "
                                            " enum member");
 }
```

```
enum TargetPosition {
    TARGET_AT_BEGINNING,
    TARGET_AT_MIDDLE,
    TARGET_AT_END
 };

 if (TARGET_AT_BEGINNING == targetPosition) {
    offsetInWindow = 0;
 } else if (TARGET_AT_MIDDLE == targetPosition) {
    offsetInWindow = MIN (newSize - size, newSize / 2);
 } else if (TARGET_AT_END == targetPosition) {
    offsetInWindow = newSize - size;
 } else {
    _error_code = IllegalData;
    _error_msg  = tr("requested target position"
                     " is not a TargetPosition "
                                            " enum member");
 }
```

```
; в edx помещено значение управляющего выражения
; т.е. targetPosition
 cmp  edx, TARGET_AT_BEGINNING
 jne  .comp2
; код для case TARGET_AT_BEGINNING:
 jmp .switch_exit
.comp2:
 cmp  edx, TARGET_AT_MIDDLE
 jne  .comp3
; код для case TARGET_AT_MIDDLE:
 jmp .switch_exit
.comp3:
 cmp  edx, TARGET_AT_END
 jne  .default
; код для case TARGET_AT_END:
 jmp .switch_exit
.default:
; код для default:
.switch_exit:
```

```
long switch_eg(long x, long y,
long z)
{
    long w = 1;
    switch(x) {
       case 1:      // .L3
           w = y*z;
           break;
       case 2:      // .L4
           w = y/z;
       /* «проваливаемся» */
       merge:       // .L9
           w += z;
           break;
       case 3:      // .L5
           w += z;
           break;
       case 5:
       case 6:      // .L6
           w -= z;
           break;
       default:     // .L2
           w = 2;
  }
  return w;
}
```

```
switch_eg:
     push  ebp                  ;
     mov   ebp, esp             ;
     mov   eax, dword [ebp + 8]; eax = x
     cmp   eax, 6
     ; сравниваем x и 6
     ja    .L2
     ; если >u goto default
     jmp   [.L7 + 4*eax]        ; goto *JTab[x]
.L2:                            ; default
     mov   eax, 2               ; w = 2
     jmp   .L8                  ; goto done

.L5:                            ; x == 3
     mov   eax, 1               ; w = 1
     jmp   .L9                  ; goto merge

.L3:                            ; x == 1
     mov   eax, dword [ebp + 16]
                                ; z
     imul  eax, dword [ebp + 12]
                                ; w = y*z
     jmp   .L8                  ; goto done
.L4:                            ; x == 2
     mov   edx, dword [ebp + 12]
     mov   eax, edx
     sar   edx, 31
     idiv  dword [ebp + 16]     ; w = y/z

.L9:                            ; merge:
     add   eax, dword [ebp + 16]
                                ; w += z
     jmp   .L8                  ; goto done

.L6:                            ; x == 5, 6
     mov   eax, 1               ; w = 1
     sub   eax, dword [ebp + 16]
                                ; w = 1-z
.L8:                            ; done:
     pop   ebp
     ret
```