Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Архітектура комп’ютерів-1

Лабораторна робота №3

**«**[**Розробка мікропрограм виконання операцій перетворення даних в ЕОМ**](#_Toc327688567) **з мікропрограмним управлінням»**

Виконала:

студентка групи ІВ-71

Молчанова В.С.

Перевірив:

доц. Верба О. А.

Київ - 2019 р.

# Завдання за варіантом:

Номер варіанта: 741010=...1110010

При g=1: Z = X \* Y (2-ий спосіб множення)

При g=0: 

Значення операндів: X = -9, Y = -5

Форма представлення: X – ПК, Y – ПК, Z – ДК

|  |
| --- |
|  |
| *Рис. 1*: Операційна схема |
|  |
| *Рис. 2*: Ф-мікроалгоритм |

**Код програми**:

link l1: ct

link l2: rdm

link ewh: 16 \lower 15 bit of RAD

dw 0h: 1100000000001001% \ x

dw 1h: 1100000000000101% \ y

dw 2h: 0 \ z

\ g == 0: 2^3 \* x + 2^(-1) \* y

\ g == 1: x \* y

\set the basic values

accept r0: 0 \register for x values

accept r1: 0 \register for y values

accept r2: 0 \register for z values

accept r3: 0 \sign of x

accept r4: 0 \sign of y

accept r7: 0 \pointer to the memory

accept rdm\_delay: 3

{ xor nil, r0, r0; oey; ewh; } \16+ bits of RAD are set to 0

{ or nil, r7, z; oey; ewl; } \transfer value from r7 to 15- RAD bits

{ cjp rdm, cp; r; or r0, bus\_d, z; } \transfer OP to r0

{ add r7, r7, 1, z; } \inc r7

{ or nil, r7, z; oey; ewl; }

{ cjp rdm, cp; r; or r1, bus\_d, z; }

{ load rm, z; } \set RM to 0

{ and nil, r1, 1; load rm, flags; } \check first bit of r1 (g)

{ cjp rm\_z, convert\_r0; } \jump to covert\_r0 if g=1

checkr0

{ and nil, r0, 8000h; load rm, flags; }

{ cjp not rm\_z, save\_r0; }

checkr1

{ and nil, r1, 8000h; load rm, flags; }

{ cjp not rm\_z, save\_r1; }

mult

{or r0,r0,z;cjp zo,signs;}

{ and nil, r1, 1; load rm, flags; } \check first bit of r1

{ cjp rm\_z, shift; }

{add r2,r2,r0,z;}

shift

{or sll, r0, r0, z;}

{or srl, r1, r1, z;}

{cjp nz, mult;}

save\_r0

{ sub r0, r0, 49152, nz; }

{ add r3, r3, 49152, z; }

{ cjp nz, checkr1; }

save\_r1

{ sub r1, r1, 49152, nz; }

{ add r4, r4, 49152, z; }

{ cjp nz, mult; }

\r0 = r0\*8, r1 = r1/2, r1 = r0+r1, r2 = r1

func

{ add sla, r0, r0, z, z; }

{ add sla, r0, r0, z, z; }

{ add sla, r0, r0, z, z; }

{ add sla, r1, r1, z, z; }

{ sub r1, r0, r1, nz; }

{ or r2, r1, z; }

{ cjp nz, exit; }

\convert r0 to dopovnjalnyi code

\save r0 sign to r3, r0 = r0 - c000h(sign), r0 = not r0 + 1

convert\_r0

{ and nil, r0, 8000h; load rm, flags; }

{ cjp rm\_z, convert\_r1; }

{ add r3, r3, 49152, z; }

{ sub r0, r0, 49152, nz; }

{ xor r0, r0, 65535; }

{ add r0, r0, 1, z; }

{ cjp nz, convert\_r1; }

\convert r1 to dopovnjalnyi code

\save r1 sign to r4, r1 = r1 - c000h(sign), r1 = not r1 + 1

convert\_r1

{ and nil, r1, 8000h; load rm, flags; }

{ cjp rm\_z, func; }

{ add r4, r4, 49152, z; }

{ sub r1, r1, 49152, nz; }

{ xor r1, r1, 65535; }

{ add r1, r1, 1, z; }

{ cjp nz, func; }

\r3 = r3 xor r4(sign of mult), convert r2 to dc if r2<0

signs

{ xor r3, r3, r4; }

{ add r2, r2, r3, z; }

{ and nil, r2, 8000h; load rm, flags; }

{ cjp rm\_z, exit; }

{ xor r2, r2, 65535; }

{ sub r2, r2, 1, nz; }

{ cjp nz, exit; }

\exit

\write r2 to memory

exit

{ add r7, r7, 1, z; }

{ or nil, r7, z; oey; ewl; }

ll2 { cjp rdm, ll2; w; or nil, r2, z; oey; }

{}

Висновок: У даній лабораторній роботі я детально ознайомилась з архітектурою ЕОМ та її основними складовими, а також одержала базові навички написання програм на мікроасемблері.