

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
Національний аерокосмічний університет ім. М.Є. Жуковського  
«Харківський авіаційний інститут»

Факультет радіоелектроніки, комп'ютерних систем та інфокомунікацій

Кафедра комп'ютерних систем, мереж і кібербезпеки (503)

Лабораторна робота № 4

	<i>Розпакування бітових груп</i>
	(назва лабораторної роботи)
з дисципліни	<i>Архітектура комп'ютерів</i>
	(шифр)

ХАІ.503.525а.03О.123-Комп'ютерна інженерія, ПЗ №9629619

Виконав студент гр.	<u>525а</u>	<u>Литвиненко А.В.</u>
09.11.22	(№ групи)	(П.І.Б.)

\_\_\_\_\_  
(підпис, дата)

Перевірив	<u>канд. техн. наук, доцент</u>
-----------	---------------------------------

_____ (підпис, дата)	<u>В. І. Дужий</u> (П.І.Б.)
-------------------------	--------------------------------

Харків – 2022

## Варіант 5

### Задача 1

#### Частина 1. Постановка завдання

##### Умова:

#### ЗАДАНИЕ

В задании графически изображен формат 32-битового двоичного числа. В каждом поле представлено название этого поля, а под соответствующим полем - его размер в битах. Выполнить распаковку упакованных двоичных групп, учитывая следующие требования:

- название каждого поля в упакованном виде является названием переменной, содержащей это поле;
- биты упакованного поля должны располагаться в младших разрядах соответствующей переменной, в то время как старшие разряды должны содержать нулевые биты;
- для размещения каждого поля использовать стандартную битовую группу минимальной длины (байт, слово или длинное слово).

##### Вхідні дані:

5

kor	reg	mod2	reg2
12	6	5	9

#### ТРЕБУЕМЫЙ РЕЗУЛЬТАТ

Определить самостоятельно, на основании формата исходного числа. Формат результирующих переменных изобразить самостоятельно, указав сверху каждого поля нумерацию битов. Название переменных определяется названием соответствующего поля, а количество переменных - количеством битовых полей в упакованном числе.

#### Частина 2. Опис алгоритму на псевдокодi

Вести вхідні дані з упакованими полями;

Виділити з числа поле reg2 на C;

Виділити з числа поле mod2 на C;

Виділити з числа поле reg на C;

Виділити з числа поле kor на C;

Виділити з числа поле reg2 на асемблері;

Виділити з числа поле mod2 на асемблері;

Виділити з числа поле reg на асемблері;

Виділити з числа поле кор на асемблері;

Вивести значення змінних на С;

Вивести значення змінних на асемблері;

### Частина 3. Схема алгоритму

На основі постановки завдання розроблений алгоритм, представлений на рисунку 1.



Рисунок 1 - Алгоритм перетворення

## Частина 4. Розробка тестів

Таблиця 1 – Тестові набори

№	Вхідні дані	Очікуваний результат				Ціль тесту
		Kop	Reg	Mod2	Reg2	
1	FFFF FFFF	Fff	3f	1f	1ff	Усі біти 1
2	AAAA AAAA	Aaa	2a	15	aa	Чередування 1 та 0
3	5555 5555	555	15	a	155	Чередування 0 та 1
4	FFF0 0000	fff	0	0	0	Всі біти 1 в kop
5	000F C000	0	3f	0	0	Всі біти 1 в reg
6	0000 3E00	0	0	1f	0	Всі біти 1 в mod2
7	0000 01FF	0	0	0	1ff	Всі біти 1 в reg2

## Частина 5. Текст програми

Відповідно до розробленого алгоритму в середовищі Microsoft Visual Studio була написана програма, яка наведена нижче.

```
/*
File: unpack.cc
Unpacking bytes group

This program unpacking byte groups from unsigned int

Input datta:
unsigned int value, which consts byte groups:

    12 + 6 + 5 + 9

    31
kop   reg   mod2   reg2   0
  12     6     5     9
*/

#include <iostream>
#include <iomanip>

using namespace std;

unsigned long value;
unsigned char mod2, reg, mod2_a, reg_a;
unsigned short kop, reg2, kop_a, reg2_a;

int main() {
    printf("\n\t\t(C) Lytvynenko A.V., 2022");
    printf("\n\tUnpacking byte groups");

    while (1) {
        printf("\n\tUnpacking 32-bit number Value");
        printf("\nPlease, enter 8 16-bits numbers (exp, 5a9db8e4) : ");
        scanf("%x", &value);

        reg2 = value & 0x1fff;
        mod2 = (value >> 9) & 0x1f;
        reg = (value >> 14) & 0x3f;
        kop = (value >> 20) & 0xffff;
    }
}
```

```

__asm {
    mov eax,value

    mov reg2_a, ax
    and reg2_a, 0x1ff
    shr eax, 9

    mov mod2_a, al
    and mod2_a, 0x1f
    shr eax, 5

    mov reg_a, al
    and reg_a, 0x3f
    shr eax, 6

    mov kop_a, ax
    and kop_a, 0xffff
}

cout << hex
    << "Bytes group kop (C++): " << (int)kop
    << "\nBytes group reg (C++): " << (int)reg
    << "\nBytes group mod2 (C++): " << (int)mod2
    << "\nBytes group reg2 (C++): " << (int)reg2

    << "\n\nBytes group kop (Asm): " << (int)kop_a
    << "\nBytes group reg (Asm): " << (int)reg_a
    << "\nBytes group mod2 (Asm): " << (int)mod2_a
    << "\nBytes group reg2 (Asm): " << (int)reg2_a
    << endl;
}
return 0;
}

```

## Частина 6. Тестування

Результати тестування наведені в таблиці 2.

Таблиця 2 – Результати тестування

№	Вхідні дані	Очікуваний результат				Отриманий результат				Статус
		Kop	Reg	Mod 2	Reg2	Kop	Reg	Mod 2	Reg	
1	FFFF FFFF	Fff	3f	1f	1ff	Fff	3f	1f	1ff	OK
2	AAAA AAAA	Aaa	2a	15	aa	Aaa	2a	15	aa	OK
3	5555 5555	555	15	a	155	555	15	a	155	OK
4	FFF0 0000	fff	0	0	0	fff	0	0	0	OK
5	000F C000	0	3f	0	0	0	3f	0	0	OK
6	0000 3E00	0	0	1f	0	0	0	1f	0	OK
7	0000 01FF	0	0	0	1ff	0	0	0	1ff	OK

## Скриншот тестування:

```

(C) Lytvynenko A.V., 2022
Unpacking byte groups
Unpacking 32-bit number Value
Please, enter 8 16-bits numbers (exp, 5a9db8e4) : ffffffff
Bytes group kop (C++): fff
Bytes group reg (C++): 3f
Bytes group mod2 (C++): 1f
Bytes group reg2 (C++): 1ff

Bytes group kop (Asm): fff
Bytes group reg (Asm): 3f
Bytes group mod2 (Asm): 1f
Bytes group reg2 (Asm): 1ff

Unpacking 32-bit number Value
Please, enter 8 16-bits numbers (exp, 5a9db8e4) : aaaaaaaa
Bytes group kop (C++): aaa
Bytes group reg (C++): 2a
Bytes group mod2 (C++): 15
Bytes group reg2 (C++): aa

Bytes group kop (Asm): aaa
Bytes group reg (Asm): 2a
Bytes group mod2 (Asm): 15
Bytes group reg2 (Asm): aa

```

```

Unpacking 32-bit number Value
Please, enter 8 16-bits numbers (exp, 5a9db8e4) : 55555555
Bytes group kop (C++): 555
Bytes group reg (C++): 15
Bytes group mod2 (C++): a
Bytes group reg2 (C++): 155

Bytes group kop (Asm): 555
Bytes group reg (Asm): 15
Bytes group mod2 (Asm): a
Bytes group reg2 (Asm): 155

Unpacking 32-bit number Value
Please, enter 8 16-bits numbers (exp, 5a9db8e4) : 0fc000
Bytes group kop (C++): 0
Bytes group reg (C++): 3f
Bytes group mod2 (C++): 0
Bytes group reg2 (C++): 0

Bytes group kop (Asm): 0
Bytes group reg (Asm): 3f
Bytes group mod2 (Asm): 0
Bytes group reg2 (Asm): 0

Unpacking 32-bit number Value
Please, enter 8 16-bits numbers (exp, 5a9db8e4) : 03e00
Bytes group kop (C++): 0
Bytes group reg (C++): 0
Bytes group mod2 (C++): 1f
Bytes group reg2 (C++): 0

Bytes group kop (Asm): 0
Bytes group reg (Asm): 0
Bytes group mod2 (Asm): 1f
Bytes group reg2 (Asm): 0

Unpacking 32-bit number Value
Please, enter 8 16-bits numbers (exp, 5a9db8e4) : 01ff
Bytes group kop (C++): 0
Bytes group reg (C++): 0
Bytes group mod2 (C++): 0
Bytes group reg2 (C++): 1ff

Bytes group kop (Asm): 0
Bytes group reg (Asm): 0
Bytes group mod2 (Asm): 0
Bytes group reg2 (Asm): 1ff

```

Рисунок 2 – скрипти тестування

## **Висновки**

Під час цієї лабораторної роботи я навчився використовувати розпакування бітових груп з 32-бітового числа, попрактикувався у використанні логічних команд, команд сдвигів та алгоритмах розпаковки бітових груп.