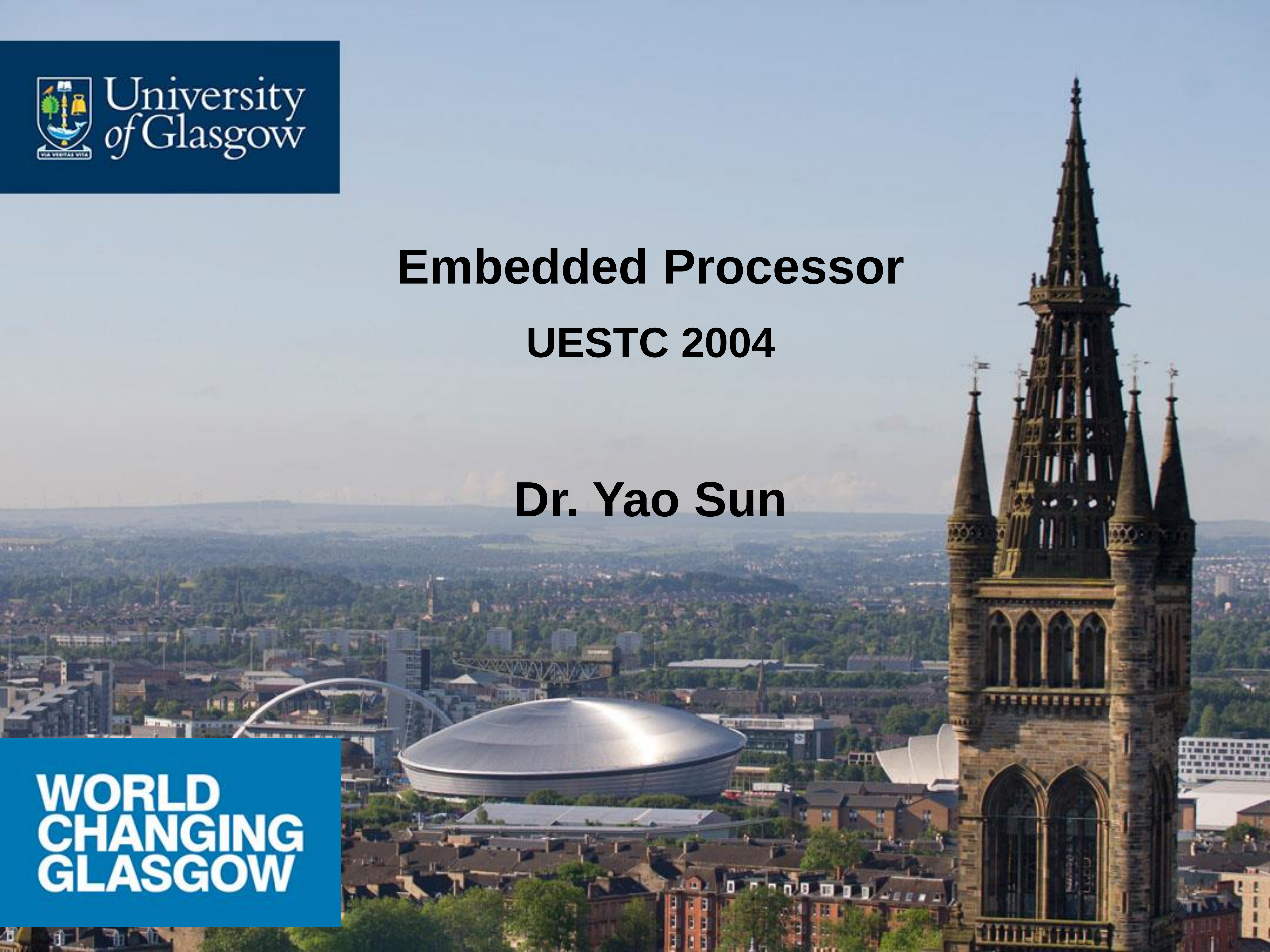


Embedded Processor

UESTC 2004

Dr. Yao Sun

WHO I AM

- Dr. Yao Sun, Lecturer with UoG UESTC College
- Yao.Sun@glasgow.ac.uk
- Main Building A1 308
- Research area Wireless Networking (5G/6G network design, ML applications, network slicing, wireless blockchain, autonomous driving, etc.)
- Email me to make an appointment

Content

- Design and Debug Interrupt Handlers (1 week)
- Bus Protocols (UART, I2C, SPI) (2 weeks)

Interrupt

Design and Debug Interrupt Handlers

- Overview the concept of Interrupt
- Design Interrupt Handlers in C
- Design Interrupt Handlers in Assembly

Interrupt

Design and Debug Interrupt Handlers

- Overview the concept of Interrupt
- Design Interrupt Handlers in C
- Design Interrupt Handlers in Assembly

Interrupt

Overview the concept of Interrupt

- **Definition:** An interrupt is the automatic transfer of software execution in response to a hardware event that is asynchronous with the current software execution.
- **Examples:**
  - Mouse moved
  - Keyboard key pressed
  - Printer out of paper
  - … …

Interrupt

Overview the concept of Interrupt (cont’d)

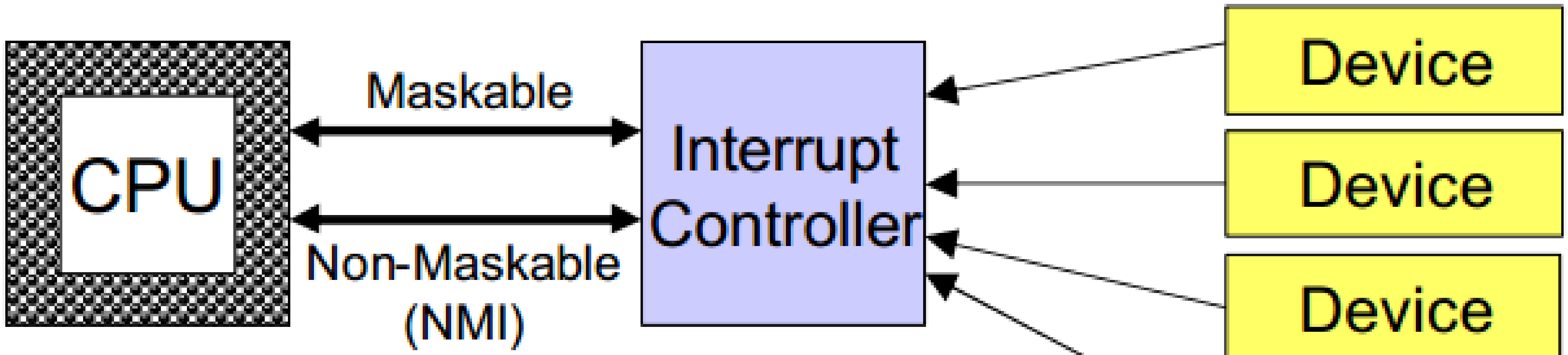


Figure Interrupt

Interrupt

Overview the concept of Interrupt (cont'd)

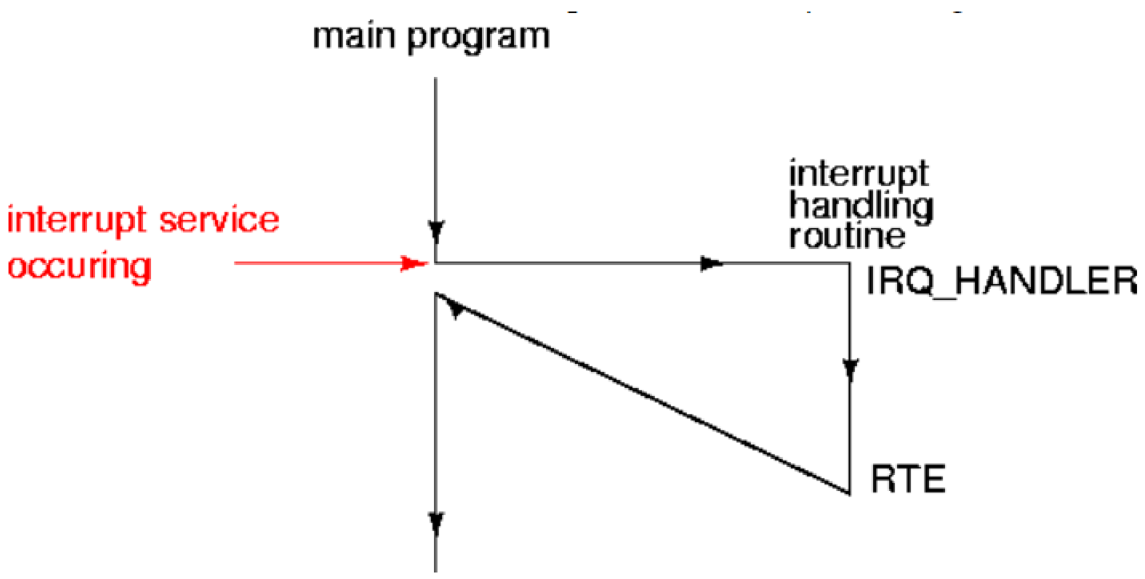


Figure Interrupt Servicing.

Interrupt

Overview the concept of Interrupt (cont'd)

- **Response to an interrupt**
  - 1) Test interrupts occur (more than one interrupts)
  - 2) Current instruction is stopped,
  - 3) Protect the status of current instruction, Eight registers are pushed on the stack,
  - 4) LR is set to 0xFFFFFFF9,
  - 5) IPSR is set to the interrupt number,
  - 6) PC is loaded with the interrupt vector

Interrupt

Overview the concept of Interrupt (cont'd)

- **Response to an interrupt (cont'd)**
  - … …
  - 7) Execute interrupt instruction/program,
  - 8) Get the interrupt return instruction,
  - 9) Go back to the main program (where to go?)

Interrupt

Design and Debug Interrupt Handlers

- Overview the concept of Interrupt
- Design Interrupt Handlers in C
- Design Interrupt Handlers in Assembly

Design Interrupt Handlers in C

Design Interrupt Handlers in C

Preparation:

1. C language Complier

   Dev C++; Visual Studio; … …

2. Basic knowledge of C

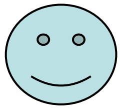
3. Process of an interrupt

Design Interrupt Handlers in C

Case Study

Let us consider a main process with an interrupt, where the main process is to input 10 random numbers with range [0, 100], and the interrupt is to input $\{11^2, 12^2, 13^2, \ldots, 20^2\}$ once the input number in the main process is out of range.

Let us implement this program in C!



Design Interrupt Handlers in C

Case Study

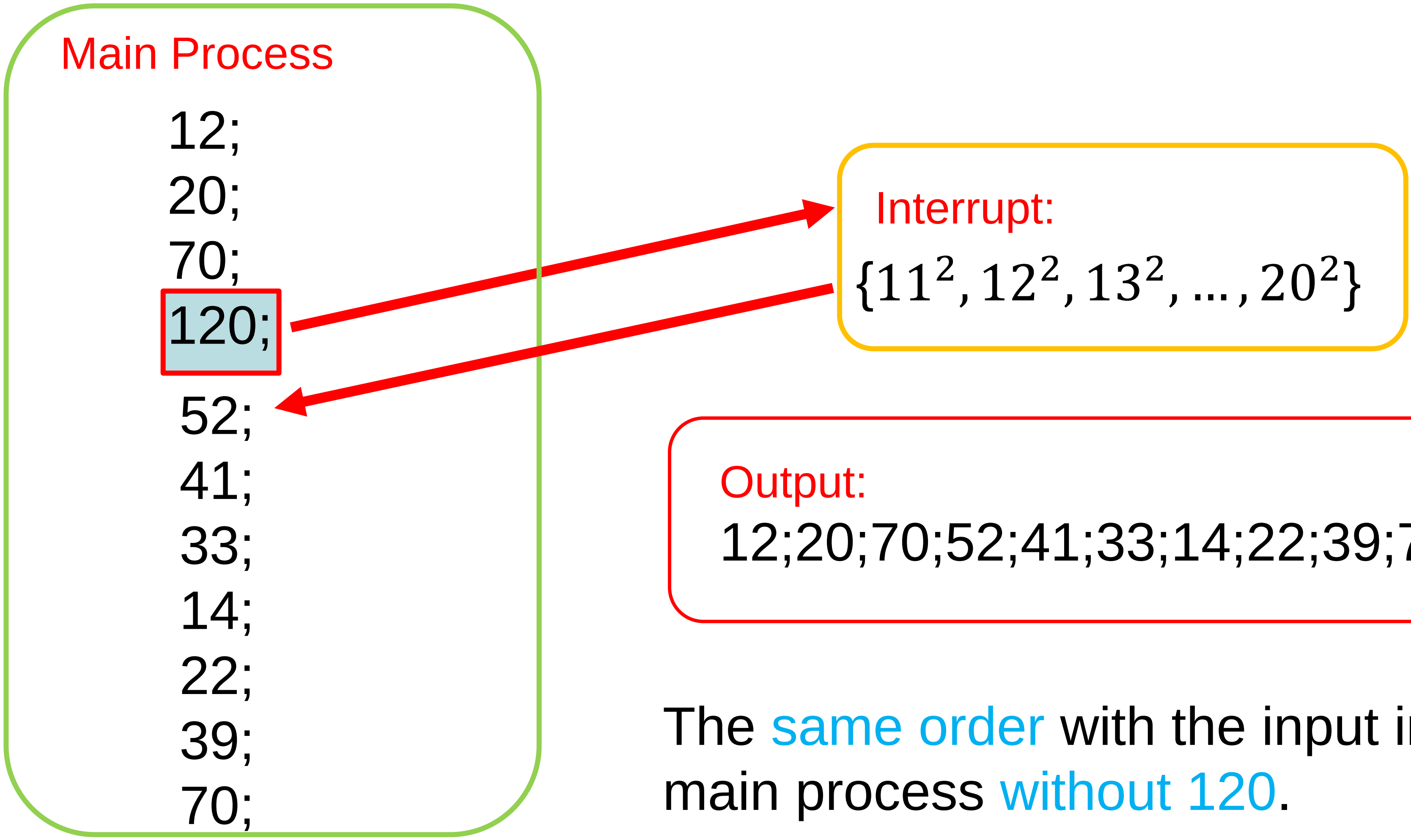
Main Process: Input 10 numbers in range [0, 100];

Interrupt Process: Input $\{11^2, 12^2, 13^2, \ldots, 20^2\}$;

Trigger Condition: The number is out of range in the main process.

Design Interrupt Handlers in C

Case Study



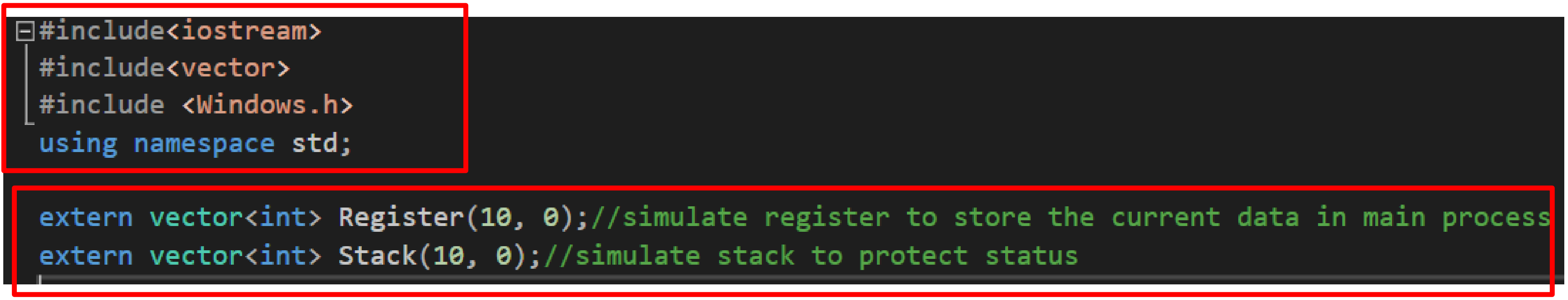
Output:
12;20;70;52;41;33;14;22;39;70

The same order with the input in main process without 120.

Design Interrupt Handlers in C

Case Study

Headers



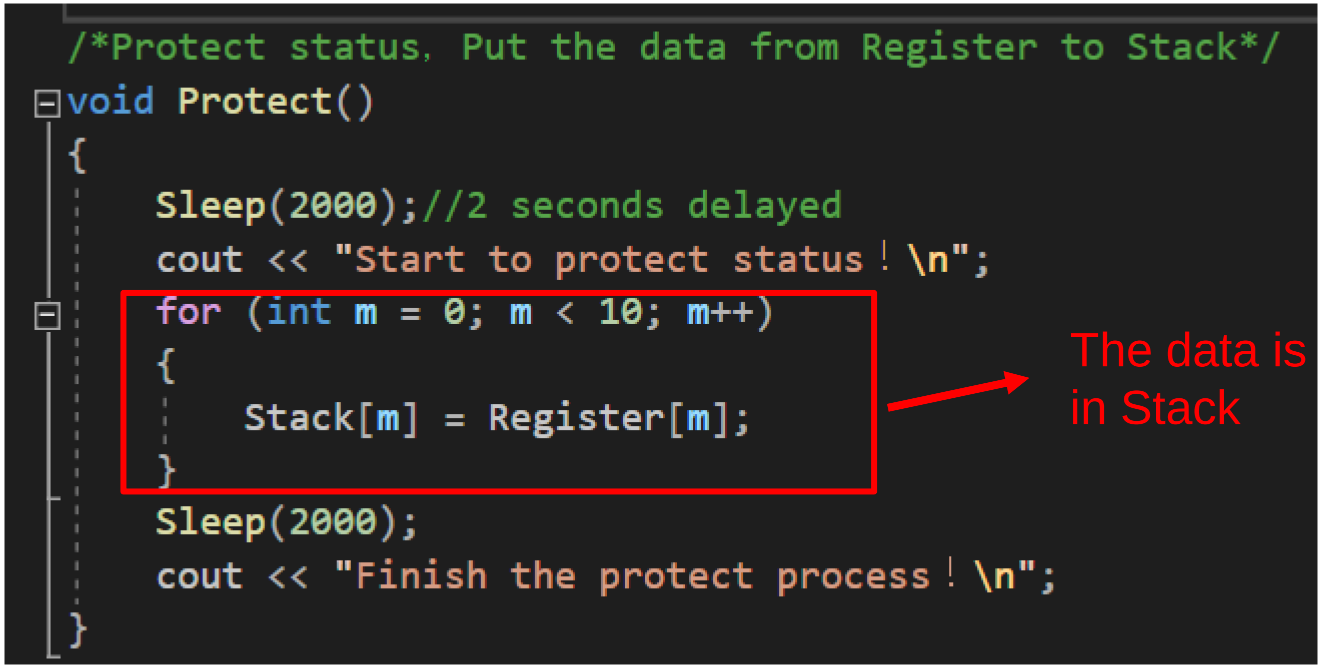
```
#include<iostream>
#include<vector>
#include <Windows.h>
using namespace std;

extern vector<int> Register(10, 0);//simulate register to store the current data in main process
extern vector<int> Stack(10, 0);//simulate stack to protect status
```

two vectors to store data in Register and Stack

Design Interrupt Handlers in C

Case Study

Protect Function



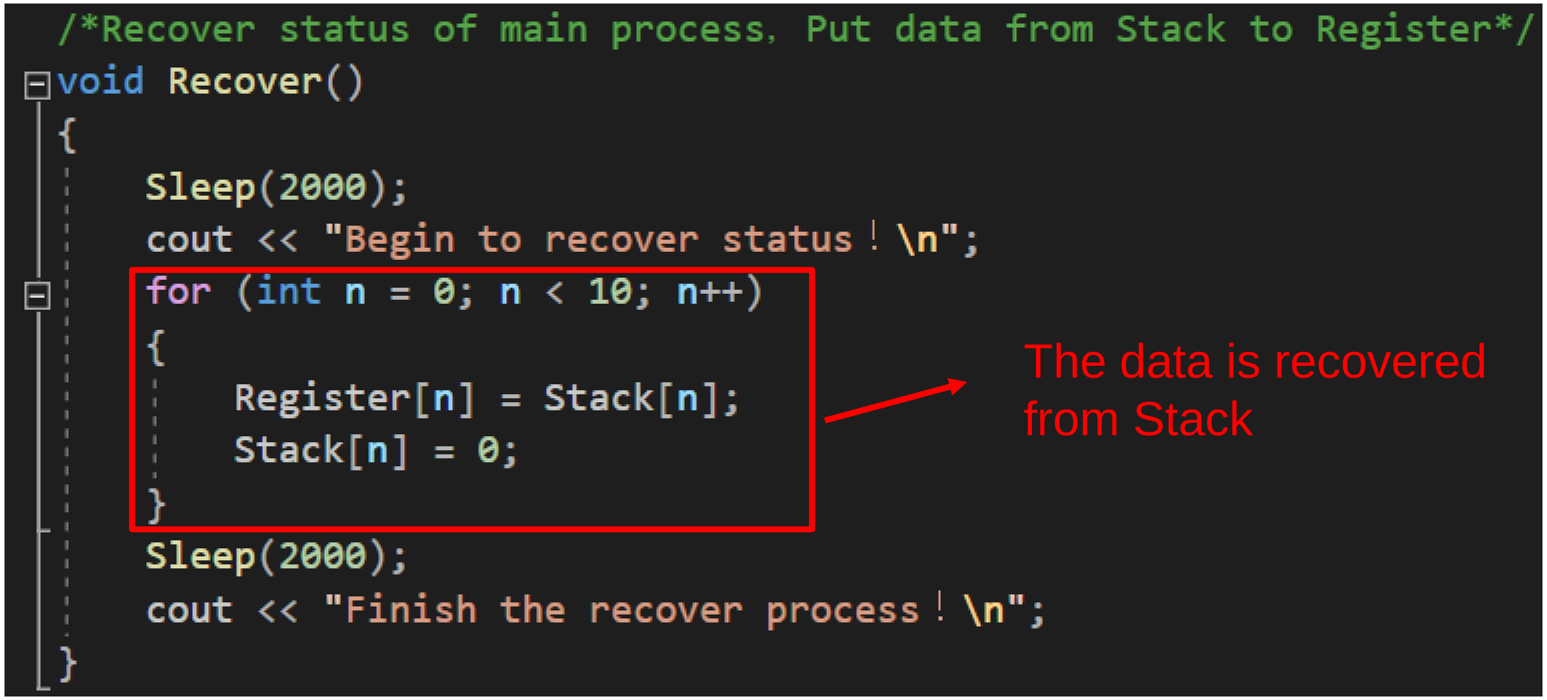
```
/*Protect status, Put the data from Register to Stack*/
void Protect()
{
    Sleep(2000);//2 seconds delayed
    cout << "Start to protect status! \n";
    for (int m = 0; m < 10; m++)
    {
        Stack[m] = Register[m];
    }
    Sleep(2000);
    cout << "Finish the protect process! \n";
}
```

The data is protected in Stack

Design Interrupt Handlers in C

Case Study

Recover Function



```
/*Recover status of main process, Put data from Stack to Register*/
void Recover()
{
    Sleep(2000);
    cout << "Begin to recover status！\n";
    for (int n = 0; n < 10; n++)
    {
        Register[n] = Stack[n];
        Stack[n] = 0;
    }
    Sleep(2000);
    cout << "Finish the recover process！\n";
}
```

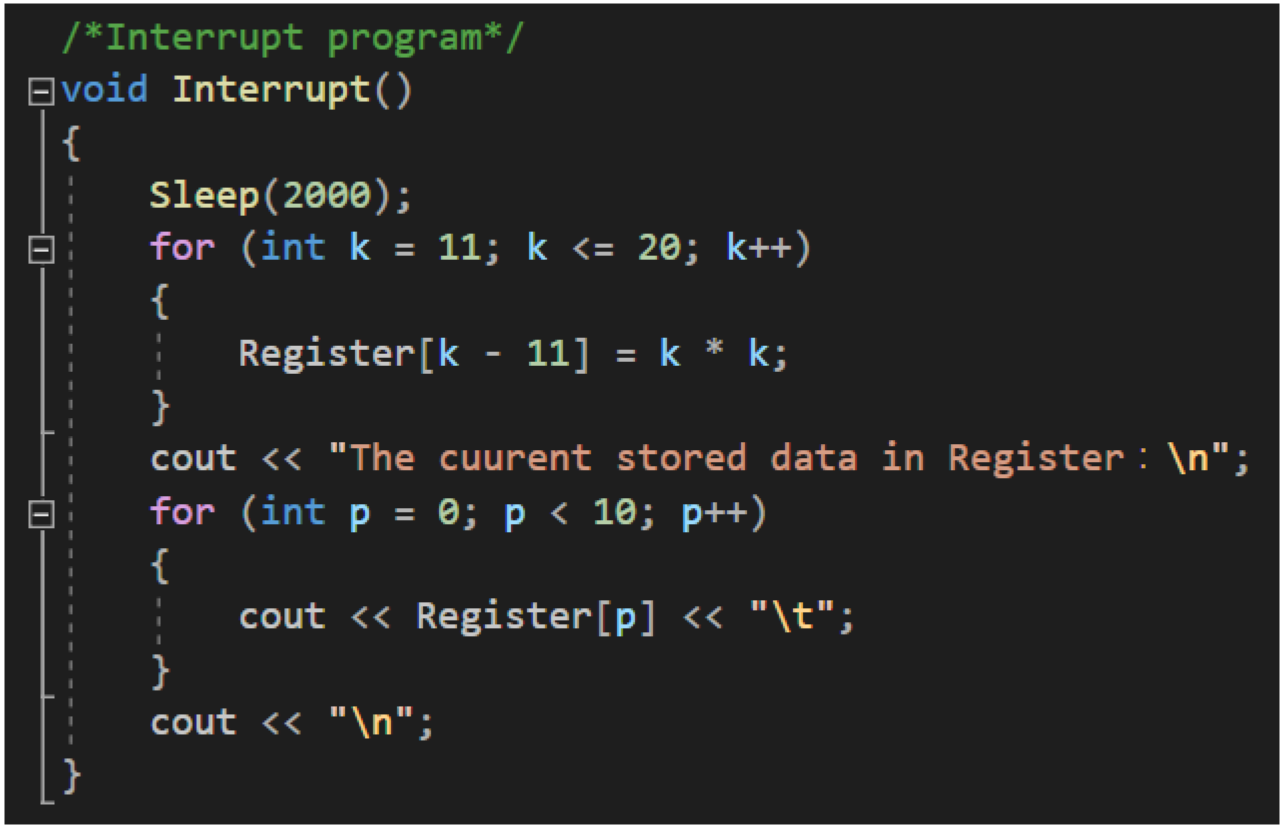
The data is recovered from Stack

University of Glasgow

Design Interrupt Handlers in C

Case Study

Interrupt Program

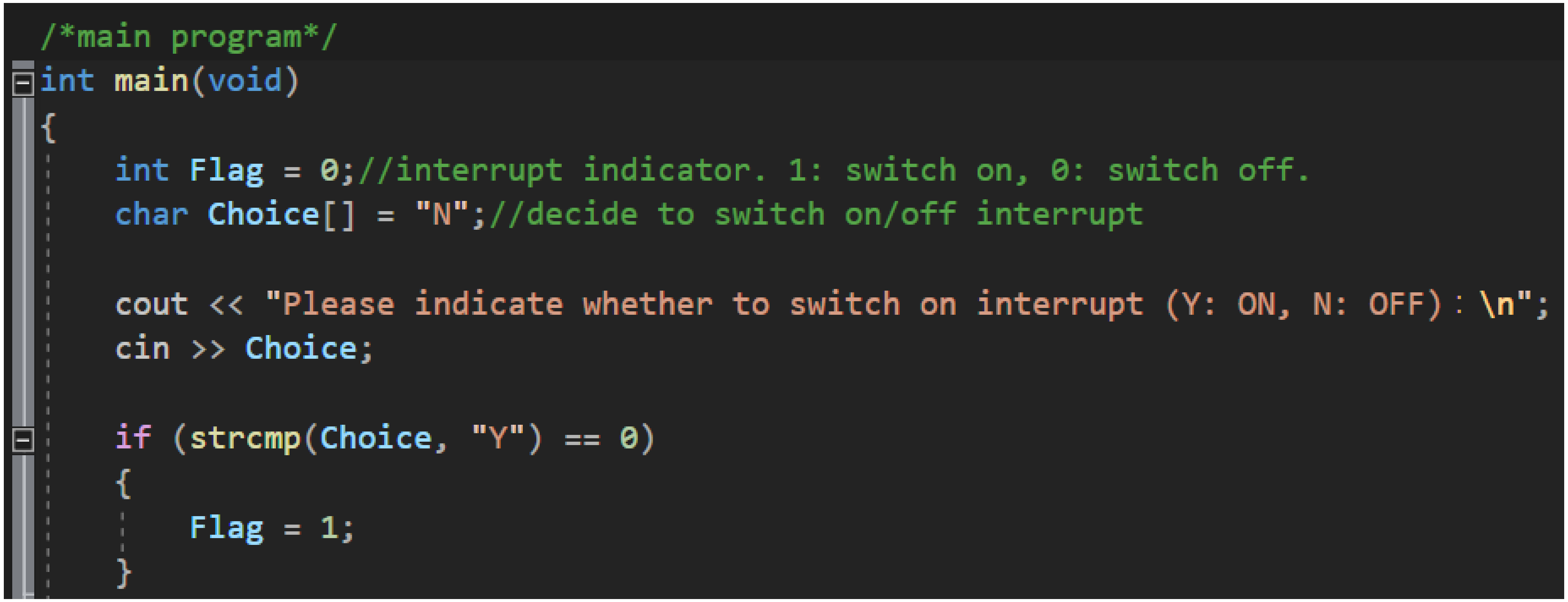


```
/*Interrupt program*/
void Interrupt()
{
    Sleep(2000);
    for (int k = 11; k <= 20; k++)
    {
        Register[k - 11] = k * k;
    }
    cout << "The cuurent stored data in Register: \n";
    for (int p = 0; p < 10; p++)
    {
        cout << Register[p] << "\t";
    }
    cout << "\n";
}
```

Design Interrupt Handlers in C

Case Study

Main Function

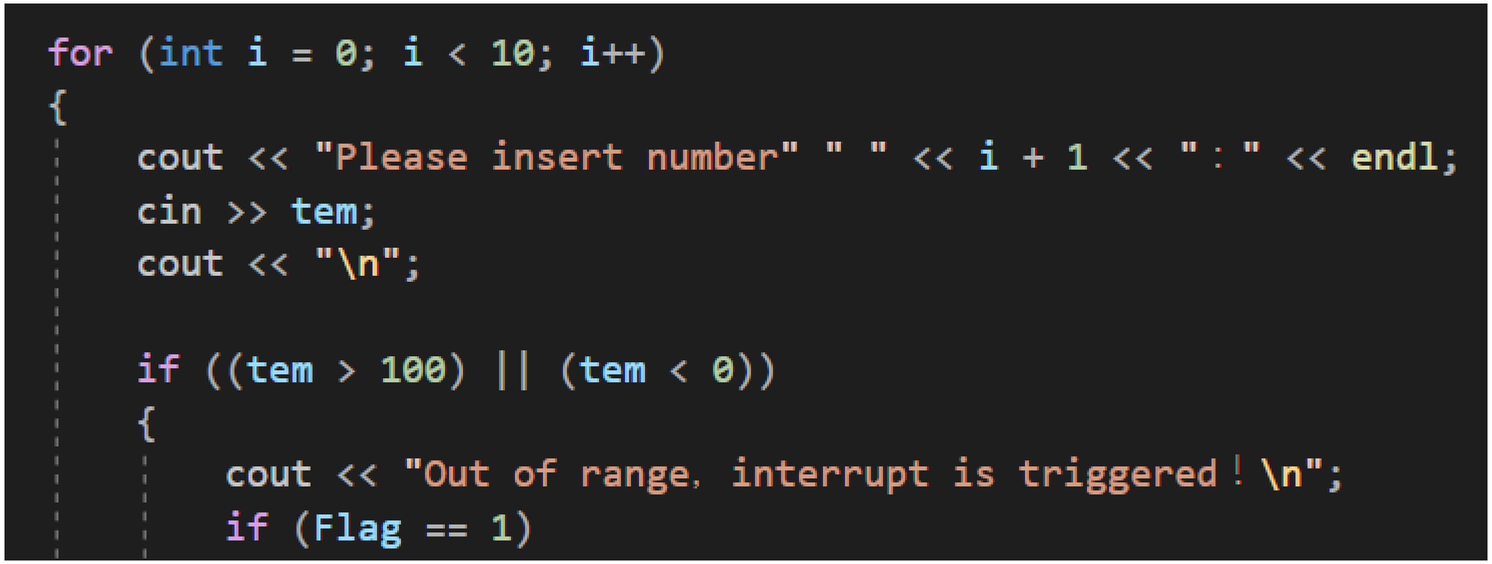


```
/*main program*/
int main(void)
{
    int Flag = 0;//interrupt indicator. 1: switch on, 0: switch off.
    char Choice[] = "N";//decide to switch on/off interrupt

    cout << "Please indicate whether to switch on interrupt (Y: ON, N: OFF): \n";
    cin >> Choice;

    if (strcmp(Choice, "Y") == 0)
    {
        Flag = 1;
    }
```

Design Interrupt Handlers in C

Case Study

Main Function (cont'd)



```
for (int i = 0; i < 10; i++)
{
    cout << "Please insert number" " " << i + 1 << ":" << endl;
    cin >> tem;
    cout << "\n";

    if ((tem > 100) || (tem < 0))
    {
        cout << "Out of range, interrupt is triggered! \n";
        if (Flag == 1)
```

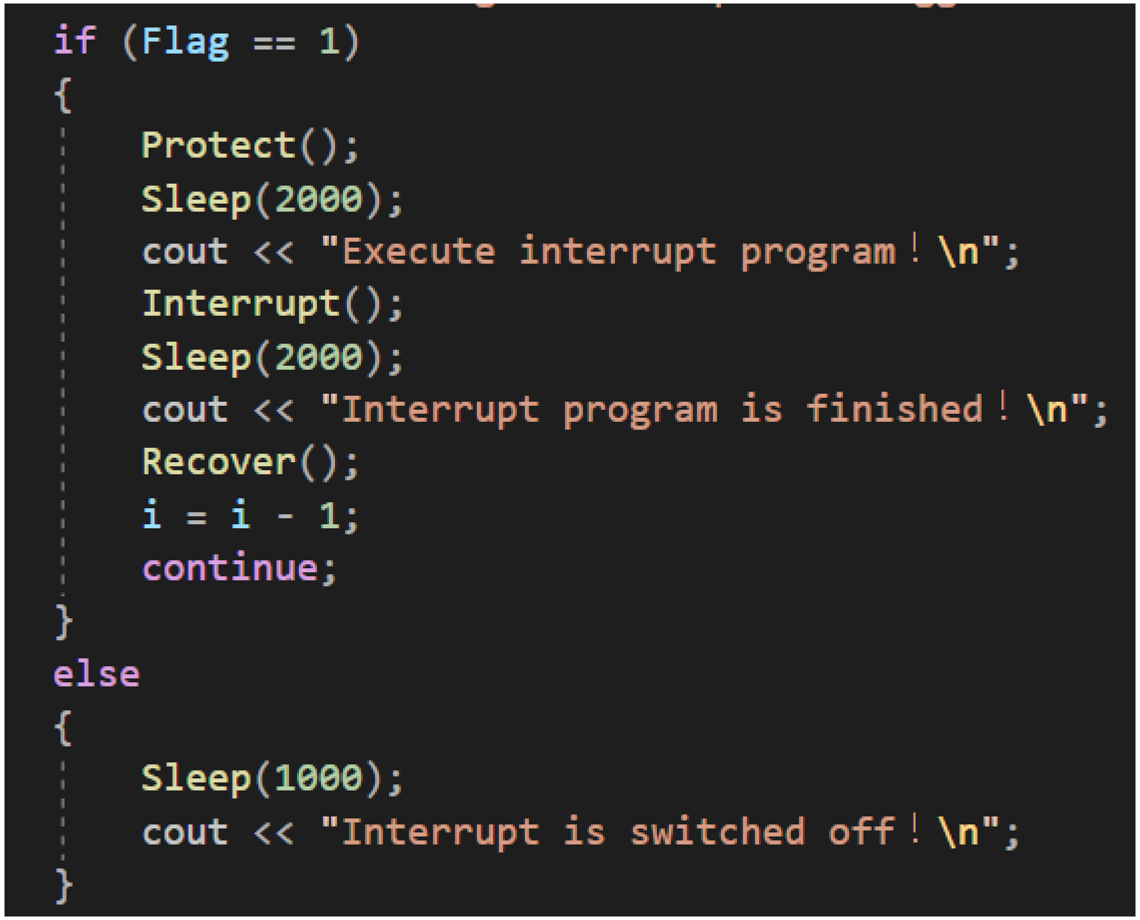
Interrupt occurs

Design Interrupt Handlers in C

Case Study

Main Function (cont'd)

Response to interrupt



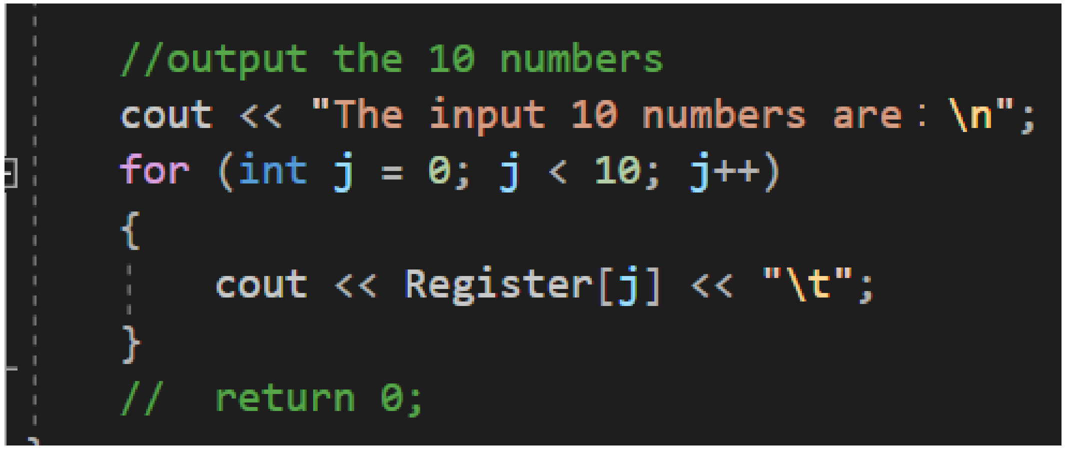
```
if (Flag == 1)
{
    Protect();
    Sleep(2000);
    cout << "Execute interrupt program！\n";
    Interrupt();
    Sleep(2000);
    cout << "Interrupt program is finished！\n";
    Recover();
    i = i - 1;
    continue;
}
else
{
    Sleep(1000);
    cout << "Interrupt is switched off！\n";
}
```

Design Interrupt Handlers in C

Case Study

Main Function (cont’d)

Output the results of main program



```
//output the 10 numbers
cout << "The input 10 numbers are : \n";
for (int j = 0; j < 10; j++)
{
    cout << Register[j] << "\t";
}
//  return 0;
```

Design Interrupt Handlers in C

Case Study

- Now let us see how this program can be run and debugged in C language complier.
- We use Microsoft Visual Studio as an example.

Interrupt

Design and Debug Interrupt Handlers

- Overview the concept of Interrupt
- Design Interrupt Handlers in C
- Design Interrupt Handlers in Assembly

Design Interrupt Handlers in Assembly

Assembly Language

- Low-level programming language
- Lots of interactions with physical devices
- Hard for humans to read an assembly program

Design Interrupt Handlers in Assembly

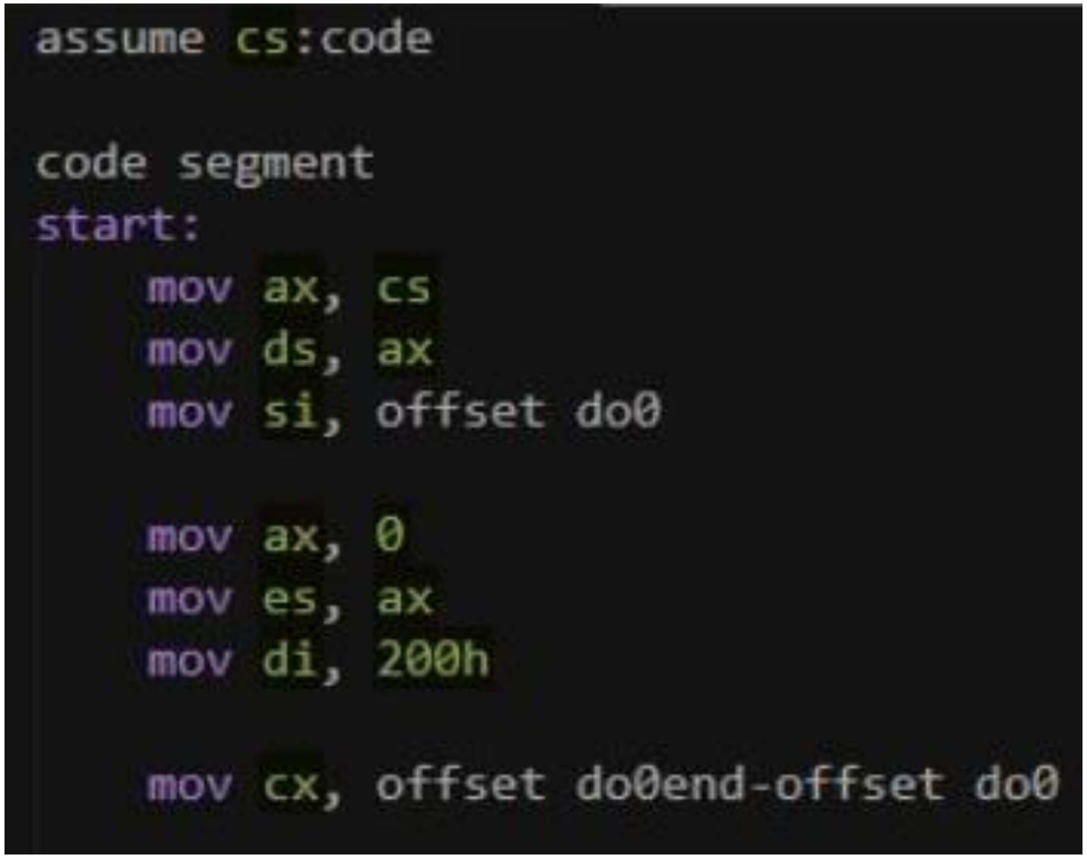
Case Study – Interrupt processing for number 0.

No. 0 Interrupt: divide overflow error, e.g. $^5/_0$.

Let us conduct an interrupt handler to handle a division overflow. Once a division overflow occurs in the system, our interrupt handler is executed, displaying a string "overflow!" in the center of the screen.

Design Interrupt Handlers in Assembly

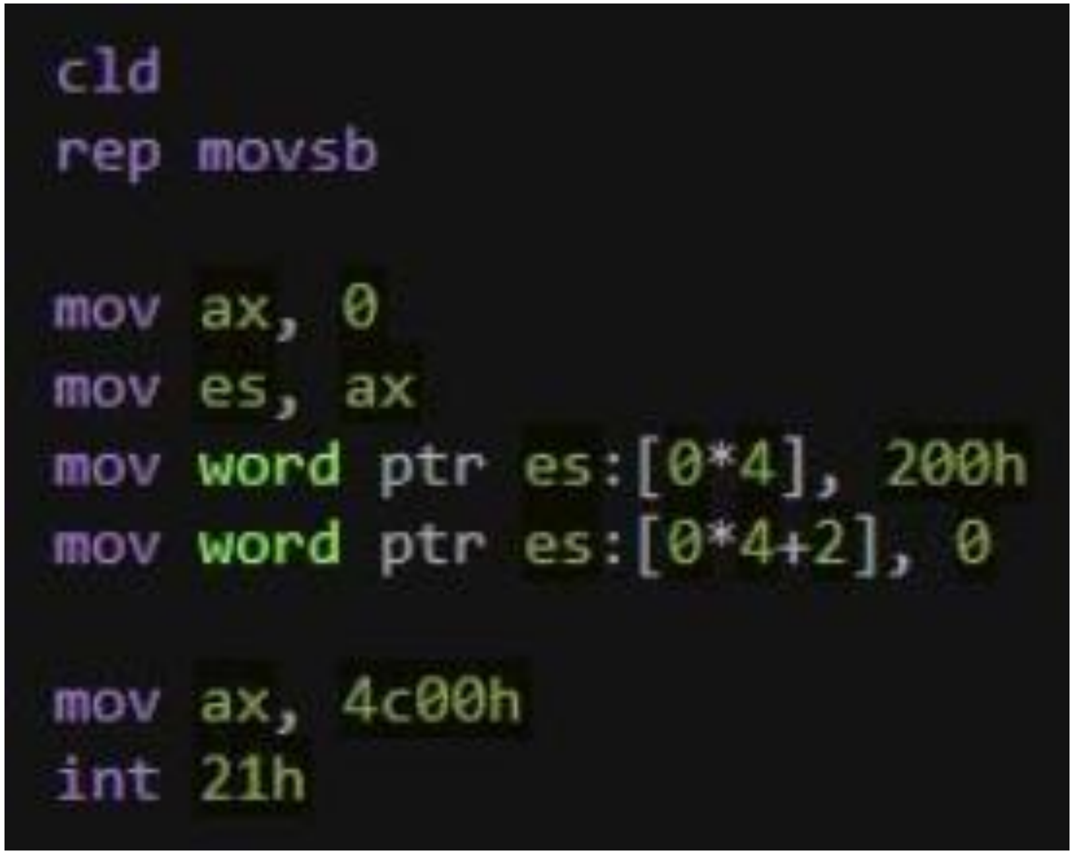
Interrupt Handler



```
assume cs:code

code segment
start:
    mov ax, cs
    mov ds, ax
    mov si, offset do0

    mov ax, 0
    mov es, ax
    mov di, 200h

    mov cx, offset do0end-offset do0
```

Design Interrupt Handlers in Assembly

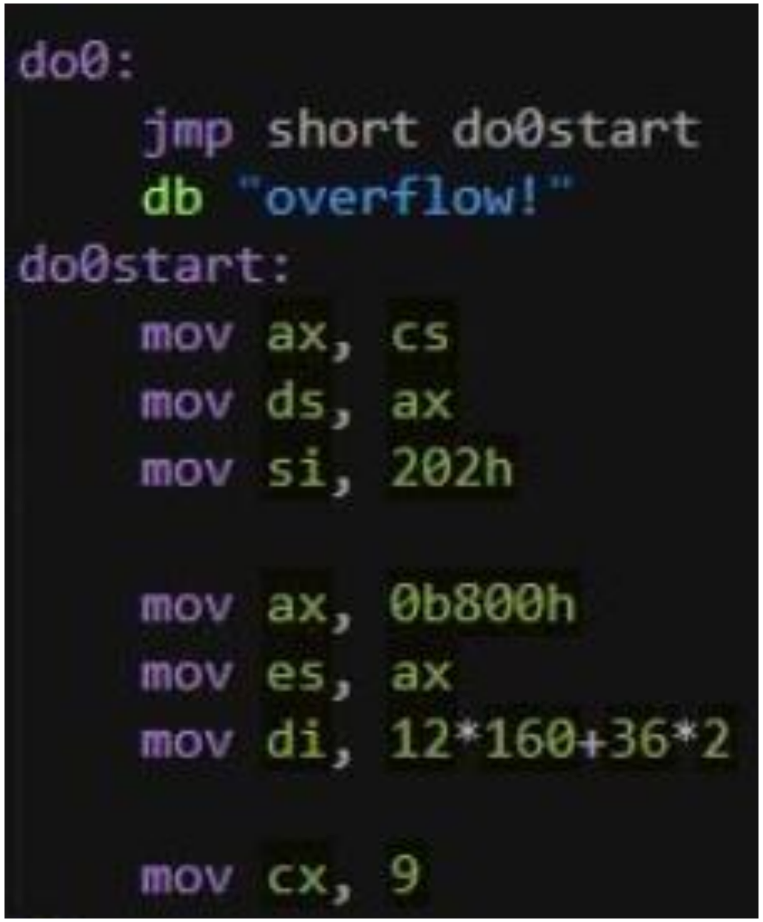
Interrupt Handler (cont'd)



```
cld
rep movsb

mov ax, 0
mov es, ax
mov word ptr es:[0*4], 200h
mov word ptr es:[0*4+2], 0

mov ax, 4c00h
int 21h
```

Design Interrupt Handlers in Assembly

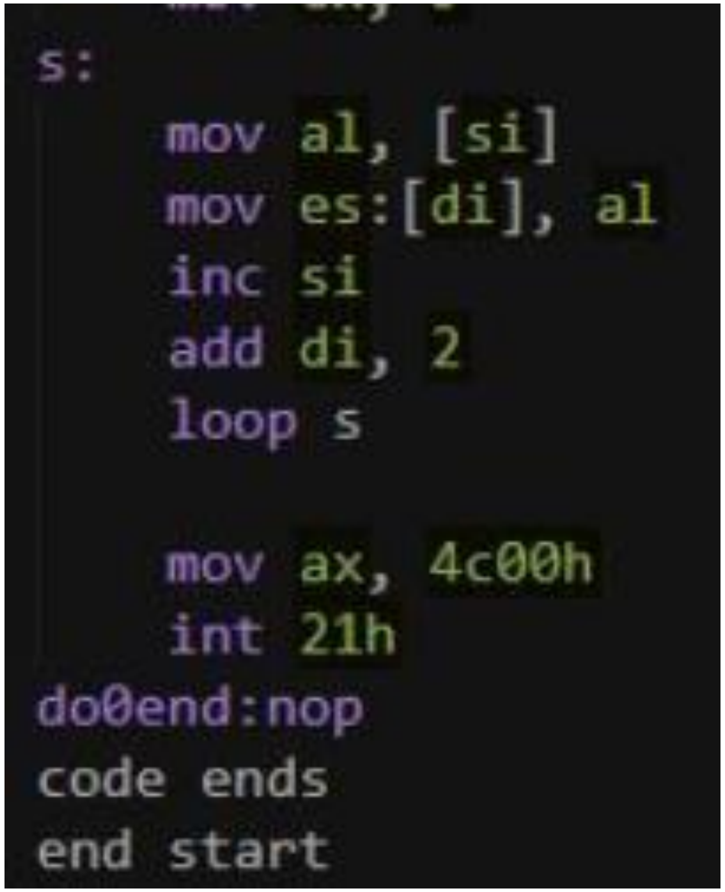
Interrupt Handler (cont'd)



```
do0:
    jmp short do0start
    db "overflow!"
do0start:
    mov ax, cs
    mov ds, ax
    mov si, 202h

    mov ax, 0b800h
    mov es, ax
    mov di, 12*160+36*2

    mov cx, 9
```

Design Interrupt Handlers in Assembly

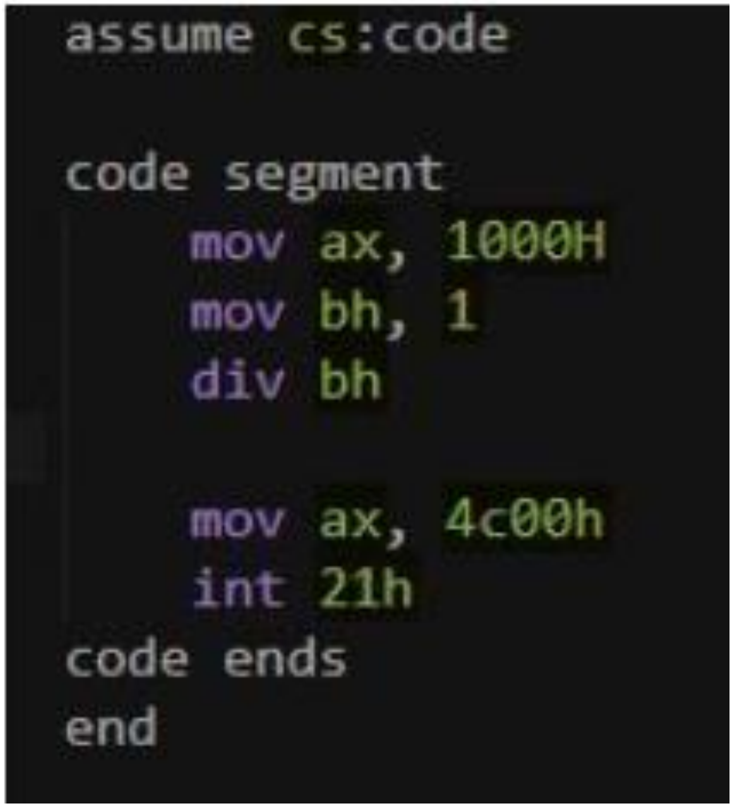
Interrupt Handler (cont'd)



```
s:
    mov al, [si]
    mov es:[di], al
    inc si
    add di, 2
    loop s

    mov ax, 4c00h
    int 21h
do0end:nop
code ends
end start
```

Design Interrupt Handlers in Assembly

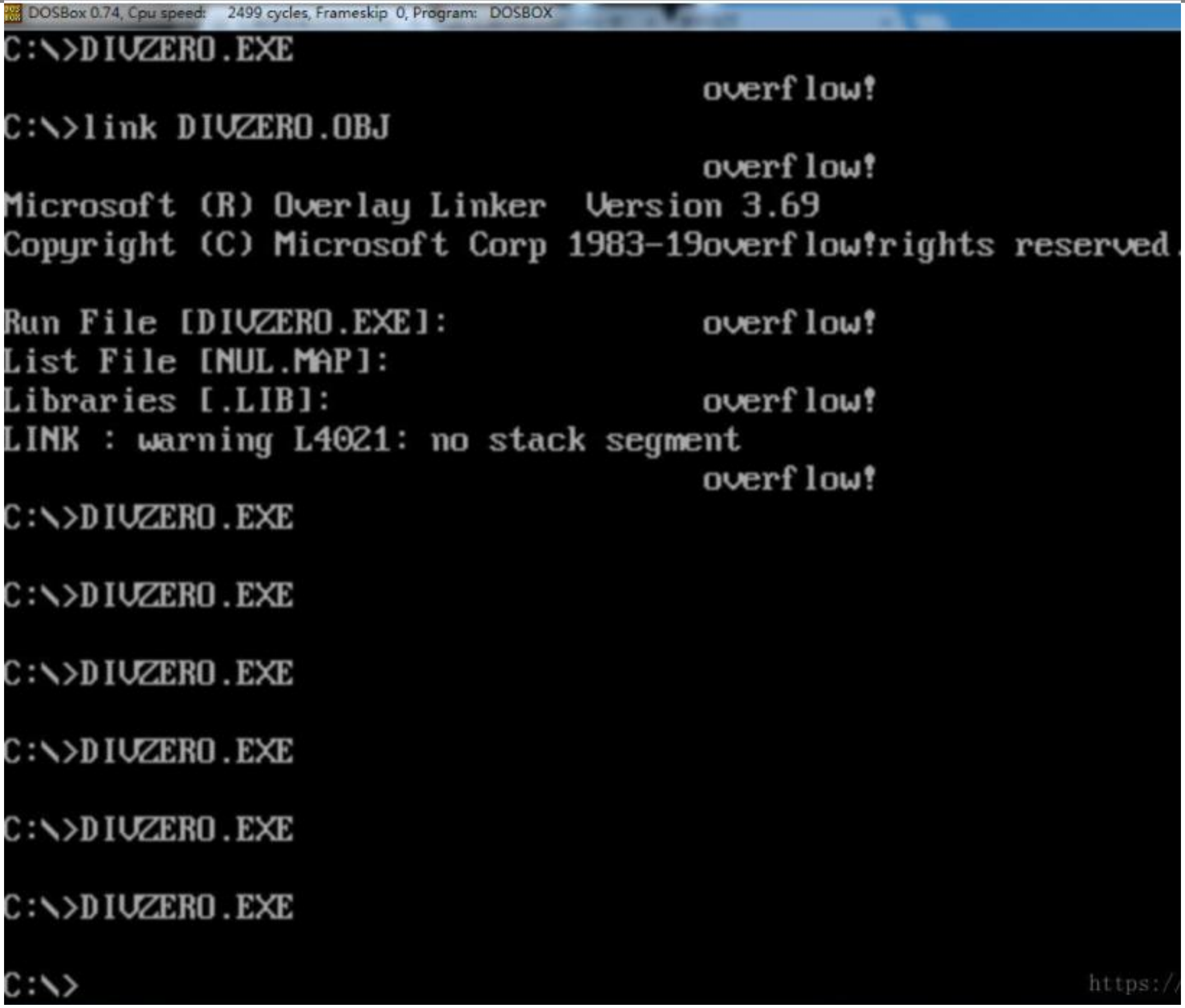
Test code



```
assume cs:code

code segment
    mov ax, 1000H
    mov bh, 1
    div bh

    mov ax, 4c00h
    int 21h
code ends
end
```

Design Interrupt Handlers in Assembly

Output


DOSBox 0.74, Cpu speed: 2499 cycles, Frameskip 0, Program: DOSBOX

```
C:\>DIVZERO.EXE
                                   overflow!
C:\>link DIVZERO.OBJ
                                   overflow!
Microsoft (R) Overlay Linker  Version 3.69
Copyright (C) Microsoft Corp 1983-19overflow!rights reserved.

Run File [DIVZERO.EXE]:            overflow!
List File [NUL.MAP]:
Libraries [.LIB]:                  overflow!
LINK : warning L4021: no stack segment
                                   overflow!
C:\>DIVZERO.EXE

C:\>DIVZERO.EXE

C:\>DIVZERO.EXE

C:\>DIVZERO.EXE

C:\>DIVZERO.EXE

C:\>DIVZERO.EXE

C:\>                                                         https://
```

University of Glasgow

THANKS!

WORLD CHANGING GLASGOW