Computer Architecture Course: IT089IU

International University – VNU HCM Date: March 2021

Dr. Le Hai Duong & Dr. Ly Tu Nga Time: 6 hours

**Laboratory Session 4**

# **Procedure (70pts)**

1. **Exercise 1: Int2Hex Converter. (45pts)**

**Write a program that**

* 1. Read in **ONE** unsigned integer in the range 0 to 15. Print out that number in hexadecimal. For example, given the input 13, print out 0xD. (**lab4\_1\_1.s**)
  2. Modify the previous assembly, create a procedure printHex(int num). This procedure takes in a number and print it out in hexadecimal. (**lab4\_1\_2.s**)
  3. Modify the previous assembly so that it can print out hexadecimal of any 32-bit integer input. For example, read in number 546263, print out 0x855D7. (**lab4\_1\_3.s**)

1. **Exercise 2: Fibonacci number (25 pts)**

The Fibonacci series is defined as:

1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ...

Each term in the series is the sum of the preceding two terms. So, for example, the term 13 is the sum of the terms 5 and 8.

2.1 Write the program as a counting loop that terminates when the first 100 terms of the series have been computed. Use a register for the current term and a register for the previous term. Each execution of the loop computes a new current term and then copies the old current term to the previous term register. Print out the series. (Use no function) (**lab4\_2\_1.s**)

2.2 Rewrite the program using function

void fibonacci(int numOfTerms); // this is C prototype

This function print out the Fibonacci series with one input is the number of terms numOfTerms. (**lab4\_2\_2.s**)

2.3 Rewrite the program using recursion. The fib function should compute the nth term as such: (**lab4\_2\_3.s**)

int fib (int N) {

if N > 1

return fib(N – 1) + fib(N – 2);

else

return 1;

}

# **II. MSP430 (30pts)**

Given a sample code to control LED **under Timer and interrupt** in MSP430 as follows

|  |  |  |
| --- | --- | --- |
| **No.** | **Sample codes** | **Comments/Results/Functions** |
| **1.**  **2**  **3**  **4**  **5**  **6**  **7**  **8**  **9**  **10**  **11**  **12**  **13**  **14**  **15**  **16**  **17**  **18**  **19**  **20**  **21**  **22**  **23**  **24**  **25**  **26**  **27**  **28**  **29**  **30**  **31**  **32**  **33**  **34**  **35**  **36**  **37**  **38**  **39**  **40**  **41**  **42**  **43**  **44**  **45**  **46**  **47**  **48**  **49**  **50**  **51**  **52**  **53** | **#include** <msp430.h>  **void** **Configure\_Clock**(**void**);  **void** **Configure\_Timer**(**void**);  **void** **Configure\_IO**(**void**);  **void** **main**(**void**) {  WDTCTL = WDTPW | WDTHOLD;  Configure\_Clock();  Configure\_IO();  Configure\_Timer();  \_BIS\_SR(GIE);  **while**(1)  {  \_no\_operation();  }  }  **#pragma** vector=TIMER0\_A1\_VECTOR  **\_\_interrupt** **void** **Timer\_A\_1**(**void**)  {  **switch**(TA0IV)  {  **case** 2: **break**;  **case** 4: **break**;  **case** 10: P1OUT ^=BIT0;  **break**;  }  }  **void** **Configure\_Clock**(**void**)  {  **if** (CALBC1\_1MHZ==0xFf)  {  **while** (1);  }  DCOCTL=0;  BCSCTL1=CALBC1\_1MHZ;  DCOCTL=CALDCO\_1MHZ;  BCSCTL2 |=SELM\_0;  }  **void** **Configure\_Timer**(**void**)  {  TA0CCR0 =50000;  TA0CTL |=TASSEL\_2+MC\_1+TAIE;  }  **void** **Configure\_IO**(**void**){  P1DIR |= BIT0+BIT6;  P1OUT &= ~(BIT0+BIT6);  } | Configure clock for system  Configure timer for system  Configure Input Output for system    // Stop watchdog timer  //enter global interrupt  // Timer A0 interrupt service routine Overflow  // CCR1 not used  //CCR2 not used  //Clock is 1MHz  // assign the counter of Timer  // assign up mode for Timer  // turn off all leds |

**\*Evaluation memory allocation:**

**\*Observe the register of Timer0\_A3 in Core registers:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Before** | **After** | **Functions** |
| **TA0CTL** |  |  |  |
| **TA0CCTL0** |  |  |  |
| **TA0CCTL1** |  |  |  |
| **TA0CCTL2** |  |  |  |
| **TA0R** |  |  |  |

According to the Table above, the objective to **Timer 0** is:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**\*Observe the register of Interrupt Flag in Core registers:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Before** | **After** | **Functions** |
| **P1IFG** |  |  |  |
| **P2IFG** |  |  |  |

According to the Table above, the objective to Interrupt is: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**MIPS of sample code:**

|  |  |  |
| --- | --- | --- |
| **No.** | **C code** | **MIPS code** |
| **1.**  **2**  **3**  **4**  **5**  **6**  **7**  **8**  **9**  **10**  **11**  **12**  **13**  **14**  **15**  **16**  **17**  **18**  **19**  **20**  **21**  **22**  **23**  **24**  **25**  **26**  **27**  **28**  **29**  **30**  **31**  **32**  **33**  **34**  **35**  **36**  **37**  **38**  **39**  **40**  **41**  **42**  **43**  **44**  **45**  **46**  **47**  **48**  **49**  **50**  **51**  **52**  **53** |  |  |

According to the Table above, please draw the flow chart via Computer Architecture to call these functions: **Configure\_Timer, Configure\_IO, Timer\_A\_1, Configure\_Clock**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Please, compare this sample code without using Timer and Interrupt (Sample code of Lab 1)**

**Reference:**

1. <https://en.wikibooks.org/wiki/MIPS_Assembly/Pseudoinstructions>
2. <https://courses.missouristate.edu/KenVollmar/MARS/Help/SyscallHelp.html>
3. <https://www.assemblylanguagetuts.com/mips-assembly-programming-tutorials/#MIPS_Data_Types>
4. <https://en.wikibooks.org/wiki/MIPS_Assembly/Arithmetic_Instructions>
5. <https://gab.wallawalla.edu/~curt.nelson/cptr280/lecture/mips%20arithmetic%20instructions.pdf>