

# Design Assignment 1

---

**DO NOT REMOVE THIS PAGE DURING SUBMISSION:**

The student understands that all required components should be submitted in complete for grading of this assignment.

NO	SUBMISSION ITEM	COMPLETED (Y/N)	MARKS (/MAX)
1	COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS		
2.	INITIAL CODE OF TASK 1/A		
3.	INCREMENTAL / DIFFERENTIAL CODE OF TASK 2/B		
3.	INCREMENTAL / DIFFERENTIAL CODE OF TASK 3/C		
3.	INCREMENTAL / DIFFERENTIAL CODE OF TASK 4/D		
3.	INCREMENTAL / DIFFERENTIAL CODE OF TASK 5/E		
4.	SCHEMATICS		
5.	SCREENSHOTS OF EACH TASK OUTPUT		
5.	SCREENSHOT OF EACH DEMO		
6.	VIDEO LINKS OF EACH DEMO		
7.	GOOGLECODE LINK OF THE DA		

1.	INITIAL CODE OF Task1		
----	-----------------------	--	--

```

;Code segment that puts 300 numbers on to the stack
.def COUNT=r25                ;counter
.def dividend=r22              ;dividend register
.def number=r12                ;number to be added is divided
.def SUM_5H=r23                ;high of sum of 5
.def SUM_5L=r24                ;low of sum of 5
.def OVERFLOW=r7               ;overflow register for sum
.macro STACK
ldi @0, high(@1)
out SPH, @0
ldi @0, low(@1)
out SPL, @0
.endmacro
STACK r16, RAMEND
ldi XH, high(RAMEND/2)         ;set X pointer to high bits of middle of ramend
ldi XL, low(RAMEND/2)          ;set X pointer to low bits of middle of ramend
ldi COUNT, 0                   ;set counter to 0
loop:    ;loop to store numbers in to RAMEND/2 location

```

2.	INITIAL CODE OF Task2		
----	-----------------------	--	--

```

;Code segment that parses the numbers and check division by 5
ldi XH, high(RAMEND/2)
ldi XL, low(RAMEND/2)
ldi YH, high(RAMEND/2)
ldi YL, low(RAMEND/2)
ldi ZH, high(RAMEND/2)
ldi ZL, low(RAMEND/2)
again:
    ld number, Z+    ;loads number in to the number var
    ld dividend, X+  ;loads number to the dividend to be divided

division5:
;loop to divide number by 5
sum_5:
;calculates the sum for division by 5
add SUM_5L, number
brvs ovr_flw5

```

3.	INITIAL CODE OF Task3		
----	-----------------------	--	--

```

;Code segment to set overflow register
ovr_flw5:
;both labels will set overflow register is the sum is greater than 8 bits
ldi r17, 0x08
mov OVERFLOW, r17 ;copies r17 to OVERFLOW register and set bit 5
subi SUM_5H, -1

```

4.	Complete code		
----	---------------	--	--

```

; cpe301Assignment1.asm
;
; Created: 2/23/2018 1:25:41 PM
; Author : YKengne
;

;Code segment that puts 300 numbers on to the stack

;Code segment that puts 300 numbers on to the stack
.def COUNT=r25                ;counter
.def dividend=r22             ;dividend register
.def number=r12               ;number to be added is divided
.def SUM_5H=r23               ;high of sum of 5
.def SUM_5L=r24               ;low of sum of 5
.def OVERFLOW=r7              ;overflow register for sum
.macro STACK
ldi @0, high(@1)
out SPH, @0
ldi @0, low(@1)
out SPL, @0
.endmacro
STACK r16, RAMEND
ldi XH, high(RAMEND/2)        ;set X pointer to high bits of middle of ramend
ldi XL, low(RAMEND/2)         ;set X pointer to low bits of middle of ramend
ldi COUNT, 0                  ;set counter to 0
loop:    ;loop to store numbers in to RAMEND/2 location

;*****

;Code segment that parses the numbers and check division by 5
ldi XH, high(RAMEND/2)
ldi XL, low(RAMEND/2)
ldi YH, high(RAMEND/2)
ldi YL, low(RAMEND/2)
ldi ZH, high(RAMEND/2)
ldi ZL, low(RAMEND/2)
again:
    ld number, Z+    ;loads number in to the number var
    ld dividend, X+  ;loads number to the dividend to be divided

division5:
;loop to divide number by 5
sum_5:
;calculates the sum for division by 5
add SUM_5L, number
brvs ovr_flw5

;*****
;Code segment to set overflow register
ovr_flw5:
;both labels will set overflow register is the sum is greater than 8 bits
ldi r17, 0x08
mov OVERFLOW, r17 ;copies r17 to OVERFLOW register and set bit 5
subi SUM_5H, -1

```

5.	Screenshots of each task		
----	--------------------------	--	--

## Task1

The screenshot displays the Proteus 8.0 SP3 software interface, showing the pin configuration and simulation results for the PIC16C84 microcontroller.

**Top Window: Pin Configuration**

Name	Value
Program Counter	0x00000000
Stack Pointer	0x00FF
X Register	0x0000
Y Register	0x0000
Z Register	0x0000
Status Register	0x0000
Cycle Counter	3
Frequency	16.000 MHz
Stop Watch	0.10 us
<b>Registers</b>	
R00	0x00
R01	0x00
R02	0x00
R03	0x00
R04	0x00
R05	0x00
R06	0x00
R07	0x00
R08	0x00
R09	0x00
R10	0x00
R11	0x00
R12	0x00
R13	0x00
R14	0x00
R15	0x00
R16	0xFF
R17	0x00
R18	0x00
R19	0x00
R20	0x00
R21	0x00
R22	0x00
R23	0x00
R24	0x00
R25	0x00
R26	0x00
R27	0x00
R28	0x00
R29	0x00
R30	0x00
R31	0x00

**Bottom Window: Simulation Results**

The bottom window shows the simulation results for the PIC16C84 microcontroller. The 'Registers' tab is selected, displaying the values of various registers.

Name	Value
Program Counter	0x00000000
Stack Pointer	0x00FF
X Register	0x0000
Y Register	0x0000
Z Register	0x0010
Status Register	0x0000
Cycle Counter	181
Frequency	16.000 MHz
Stop Watch	11.11 us
<b>Registers</b>	
R00	0x00
R01	0x00
R02	0x00
R03	0x00
R04	0x00
R05	0x00
R06	0x00
R07	0x00
R08	0x00
R09	0x00
R10	0x00
R11	0x00
R12	0x00
R13	0x00
R14	0x00
R15	0x00
R16	0xFF
R17	0xFF
R18	0x00
R19	0x00
R20	0x00
R21	0x00
R22	0x00
R23	0x00
R24	0x00
R25	0x00
R26	0x00
R27	0x00
R28	0x00
R29	0x00
R30	0x19
R31	0x19

## Task2

The screenshot displays four windows from a logic analyzer, organized in a 2x2 grid. Each window has a title bar with a close button (X) and a refresh button (F5).

- Top-Left Window (Processor Status):**
  - Name:** Processor Status
  - Value:** 0x00000004
  - Program Counter:** 0x00000004
  - Stack Pointer:** 0x00FF
  - X Register:** 0x0000
  - Y Register:** 0x0000
  - Z Register:** 0x0000
  - Status Register:** 0x0000
  - Cycle Counter:** 3
  - Frequency:** 16.000 MHz
  - Stop Watch:** 0.19 µs
  - Registers:** A list of registers (R00 to R31) with values mostly 0x00.
- Top-Right Window (Memory 4):**
  - Name:** Memory 4
  - Value:** 0x00000004
  - Program Counter:** 0x00000004
  - Stack Pointer:** 0x00FF
  - X Register:** 0x0000
  - Y Register:** 0x0000
  - Z Register:** 0x0000
  - Status Register:** 0x0000
  - Cycle Counter:** 3
  - Frequency:** 16.000 MHz
  - Stop Watch:** 0.19 µs
  - Registers:** A list of registers (R00 to R31) with values mostly 0x00.
- Bottom-Left Window (Processor Status):**
  - Name:** Processor Status
  - Value:** 0x00000004
  - Program Counter:** 0x00000004
  - Stack Pointer:** 0x00FF
  - X Register:** 0x0000
  - Y Register:** 0x0000
  - Z Register:** 0x0000
  - Status Register:** 0x0000
  - Cycle Counter:** 3
  - Frequency:** 16.000 MHz
  - Stop Watch:** 0.19 µs
  - Registers:** A list of registers (R00 to R31) with values mostly 0x00.
- Bottom-Right Window (Memory 4):**
  - Name:** Memory 4
  - Value:** 0x00000004
  - Program Counter:** 0x00000004
  - Stack Pointer:** 0x00FF
  - X Register:** 0x0000
  - Y Register:** 0x0000
  - Z Register:** 0x0000
  - Status Register:** 0x0000
  - Cycle Counter:** 3
  - Frequency:** 16.000 MHz
  - Stop Watch:** 0.19 µs
  - Registers:** A list of registers (R00 to R31) with values mostly 0x00.

## Task3

Processor Status	Name	Value	Memory	prog FLASH	Processor Status	Name	Value	Memory	data IRAM
	Program Counter	0x00000004		prog 0x0000 08 00 00 02 04 0f 0d .b.g.i.		Program Counter	0x00000019		data 0x0100 7f 83 87 8b 8f 93 97 .f...i.
	Stack Pointer	0x0001		prog 0x0002 1f 00 00 01 00 1f 0f .a.n.a.s		Stack Pointer	0x00ff		data 0x0107 0b 0f a7 ab af b7 .f.g.e.
	X Register	0x0000		prog 0x000e 29 e1 11 93 1c 5f 2a 18 .e..		X Register	0x0000		data 0x010e b7 bb bf c3 c7 cb cf .g.A.C.E.I
	Y Register	0x0000		prog 0x0015 95 20 30 d9 72 00 00 .00.00		Y Register	0x0000		data 0x0115 03 07 0b 0f 00 00 00 0x00...
	Z Register	0x0000		prog 0x0021 f1 00 29 01 10 01 00 0a0a...		Z Register	0x0119		data 0x011c 00 00 00 00 00 00 00 .....
	Status Register	0x00000000		prog 0x0023 04 60 00 11 91 00 00 .m..e.		Status Register	0x00000000		data 0x0120 00 00 00 00 00 00 00 .....
	Cycle Counter	0		prog 0x0026 50 00 2a 95 20 30 b5 P..0.		Cycle Counter	4794		data 0x0121 00 00 00 00 02 00 00 .....
	Frequency	16.000 Mhz		prog 0x0031 7f ff cf ff ff ff ff 0x00000000		Frequency	16.000 Mhz		data 0x0128 00 00 00 00 00 00 00 .....
	Stop Watch	0.19 us		prog 0x0038 ff ff ff ff ff ff ff 0x00000000		Stop Watch	290.00 us		data 0x0130 00 00 00 00 00 00 00 .....
	Registers			prog 0x0046 ff ff ff ff ff ff ff 0x00000000		Registers			data 0x0140 00 00 00 00 00 00 00 .....
R00	0x00			prog 0x0054 ff ff ff ff ff ff ff 0x00000000		R00	0x00		data 0x0150 00 00 00 00 00 00 00 .....
R01	0x00			prog 0x0058 ff ff ff ff ff ff ff 0x00000000		R01	0x00		data 0x0160 00 00 00 00 00 00 00 .....
R02	0x00			prog 0x0062 ff ff ff ff ff ff ff 0x00000000		R02	0x00		data 0x0170 00 00 00 00 00 00 00 .....
R03	0x00			prog 0x0066 ff ff ff ff ff ff ff 0x00000000		R03	0x00		data 0x0177 00 00 00 00 00 00 00 .....
R04	0x00			prog 0x0070 ff ff ff ff ff ff ff 0x00000000		R04	0x00		data 0x017e 00 00 00 00 00 00 00 .....
R05	0x00			prog 0x0074 ff ff ff ff ff ff ff 0x00000000		R05	0x00		data 0x0186 00 00 00 00 00 00 00 .....
R06	0x00			prog 0x0078 ff ff ff ff ff ff ff 0x00000000		R06	0x00		data 0x018c 00 00 00 00 00 00 00 .....
R07	0x00			prog 0x0082 ff ff ff ff ff ff ff 0x00000000		R07	0x00		data 0x0193 00 00 00 00 00 00 00 .....
R08	0x00			prog 0x0086 ff ff ff ff ff ff ff 0x00000000		R08	0x00		data 0x019a 00 00 00 00 00 00 00 .....
R09	0x00			prog 0x0090 ff ff ff ff ff ff ff 0x00000000		R09	0x00		data 0x01a1 00 00 00 00 00 00 00 .....
R10	0x00			prog 0x0094 ff ff ff ff ff ff ff 0x00000000		R10	0x00		data 0x01a8 00 00 00 00 00 00 00 .....
R11	0x00			prog 0x0098 ff ff ff ff ff ff ff 0x00000000		R11	0x00		data 0x01b0 00 00 00 00 00 00 00 .....
R12	0x00			prog 0x00a2 ff ff ff ff ff ff ff 0x00000000		R12	0x00		data 0x01b8 00 00 00 00 00 00 00 .....
R13	0x00			prog 0x00a6 ff ff ff ff ff ff ff 0x00000000		R13	0x00		data 0x01c4 00 00 00 00 00 00 00 .....
R14	0x00			prog 0x00b0 ff ff ff ff ff ff ff 0x00000000		R14	0x00		data 0x01c9 00 00 00 00 00 00 00 .....
R15	0x00			prog 0x00b4 ff ff ff ff ff ff ff 0x00000000		R15	0x00		data 0x01d0 00 00 00 00 00 00 00 .....
R16	0xff			prog 0x00b8 ff ff ff ff ff ff ff 0x00000000		R16	0xff		data 0x01d9 00 00 00 00 00 00 00 .....
R17	0x00			prog 0x00c2 ff ff ff ff ff ff ff 0x00000000		R17	0x00		data 0x01e0 00 00 00 00 00 00 00 .....
R18	0x00			prog 0x00c6 ff ff ff ff ff ff ff 0x00000000		R18	0x00		data 0x01e7 00 00 00 00 00 00 00 .....
R19	0x00			prog 0x00ca ff ff ff ff ff ff ff 0x00000000		R19	0x00		data 0x01f0 00 00 00 00 00 00 00 .....
R20	0x00			prog 0x00ce ff ff ff ff ff ff ff 0x00000000		R20	0x02		data 0x0200 00 00 00 00 00 00 00 .....
R21	0x00			prog 0x00d2 ff ff ff ff ff ff ff 0x00000000		R21	0x02		data 0x0208 00 00 00 00 00 00 00 .....
R22	0x00			prog 0x00d6 ff ff ff ff ff ff ff 0x00000000		R22	0x00		data 0x0217 00 00 00 00 00 00 00 .....
R23	0x00			prog 0x00da ff ff ff ff ff ff ff 0x00000000		R23	0x00		data 0x0226 00 00 00 00 00 00 00 .....
R24	0x00			prog 0x00de ff ff ff ff ff ff ff 0x00000000		R24	0x00		data 0x022d 00 00 00 00 00 00 00 .....
R25	0x00			prog 0x00e2 ff ff ff ff ff ff ff 0x00000000		R25	0x00		data 0x0234 00 00 00 00 00 00 00 .....
R26	0x00			prog 0x00e6 ff ff ff ff ff ff ff 0x00000000		R26	0x00		data 0x023b 00 00 00 00 00 00 00 .....
R27	0x00			prog 0x00ea ff ff ff ff ff ff ff 0x00000000		R27	0x00		data 0x0242 00 00 00 00 00 00 00 .....
R28	0x00			prog 0x00ee ff ff ff ff ff ff ff 0x00000000		R28	0x00		data 0x0249 00 00 00 00 00 00 00 .....
R29	0x00			prog 0x00f2 ff ff ff ff ff ff ff 0x00000000		R29	0x00		data 0x0250 00 00 00 00 00 00 00 .....
R30	0x00			prog 0x00f6 ff ff ff ff ff ff ff 0x00000000		R30	0x19		data 0x0257 00 00 00 00 00 00 00 .....
R31	0x00			prog 0x00fa ff ff ff ff ff ff ff 0x00000000		R31	0x01		data 0x025e 00 00 00 00 00 00 00 .....

2.	GITHUB LINK OF THE DA		
----	-----------------------	--	--

## Student Academic Misconduct Policy

<http://studentconduct.unlv.edu/misconduct/policy.html>

*"This assignment submission is my own, original work".*

Yannick Kengne Tatcha