1. Write an assembly language program to add 34FA 10 times and update the flag.

AREA main, CODE, READONLY

EXPORT \_\_main

ENTRY

\_\_main

LDR R1, =0X000034FA

MOV R2 ,#10

MOV R3,#0

L1 ADDS R3,R3,R1

SUBS R2,R2,#1

BNE L1

END

1. Write an assembly language program to complement 34FA 11 times and store the result in register. Check the status of the flag after execution.

AREA main, CODE, READONLY

EXPORT \_\_main

ENTRY

\_\_main

MOV R1, #0x34FA

MOV R2, #11

L1 MVN R1,R1

SUBS R2, R2 , #1

BNE L1

END

1. Write an assembly language program to calculate factorial of 10.

AREA main, CODE, READONLY

EXPORT \_\_main

ENTRY

\_\_main

MOV R1,#1

MOV R2,#10

L1 MUL R1, R2

SUBS R2,R2,#1

BNE L1

END

1. Write an assembly language program to perform logical operation on 0x0A in ARM cortex M4.

AREA main, CODE, READONLY

EXPORT \_\_main; make main visible to linker

ENTRY

\_\_main

MOV R1, #0x01

MOV R2, #0x01

ANDS R3,R1, R2

BICS R4,R1, R2

EORS R5,R1, R2

ORNS R6,R1, R2

ORRS R7,R1, R2

END

1. Write an assembly language program to perform Fibonacci series and store the output in memory location.

AREA main, CODE, READONLY

EXPORT \_\_main ; make main visible to linker

ENTRY

\_\_main

LDR R0,=0X20000000

MOV R1,#0x00

MOV R2,#0x01

MOV R3,#8

STRB R1,[R0],#1

STRB R2,[R0],#1

L1 ADD R4,R1,R2

STRB R4,[R0],#1

MOV R1,R2

MOV R2,R4

SUBS R3,R3,#1

BNE L1

END

1. Write an assembly language program to find addition of odd and even numbers stored in a memory location 0x20000000. The sum of odd and even is stored at 2 separate memory location.

AREA main, CODE, READONLY

EXPORT \_\_main ; make main visible to linker

ENTRY

\_\_main

BNE L1

LDR R3,=0x30000000

LDR R1,=0x04030201

LDR R2,=0x08070605

STR R1,[R0],#4

STR R2,[R0]

MOV R1,#10

MOV R5,#0

LDRB R4,[R4

END

1. Write an assembly language program to find out the maximum number from a string of a number AF, 3F, AD, 2A, F8, 4C,7C, 8D, 01, 2F, 3E, 7B and save the maximum number at a memory location.

AREA main,CODE,READONLY

EXPORT \_\_main

ENTRY

\_\_main

LDR R0,=0x20000000

LDR R1,=0x2AAD3FAF

LDR R2,=0x8D7C4CF8

LDR R3,=0x7B3E2F01

LDR R11,=0X20000001A

MOV R4,#0

MOV R5,#11

MOV R6,#0

STR R1,[R0,#0]

STR R2,[R0,#4]

STR R3,[R0,#8]

LDRB R4,[R0],#01

L1 LDRB R6,[R0],#01

CMP R4,R6

BCS L2

MOV R4,R6

L2 SUBS R5,R5,#1

BNE L1

STR R4,[R11]

END

1. Write an embedded c program to toggle i) Red led ii) Green led iii) Blue led using MSP432P401R.

#include "msp.h"

int main(void)

{

volatile uint32\_t i;

// Stop watchdog timer

WDT\_A->CTL = WDT\_A\_CTL\_PW | WDT\_A\_CTL\_HOLD;

// The following code toggles P1.0 port

P2->DIR |= BIT2; // Configure P1.0 as output

while(1)

{

P2->OUT ^= BIT2; // Toggle P1.0

for(i=10000; i>0; i--); // Delay

}

}

1. Write an embedded c program to perform led blink operation on RGB led P2.0, P2.1, P2.2 one after another for 2 minutes of delay using MSP432P401R.

#include "msp.h"

void delayMs(int n);

//volatile uint32\_t i;

int main(void)

{

P2->SEL0 &= ~7;

P2->SEL1 &= ~7;

//volatile uint32\_t i;

// Stop watchdog timer

//WDT\_A->CTL = WDT\_A\_CTL\_PW | WDT\_A\_CTL\_HOLD;

// The following code toggles P1.0 port

P2->DIR |=7;

//P2->OUT = BIT0|BIT1|BIT2;// Configure P1.0 as output

while(1)

{

P2->OUT = BIT0; // Toggle P1.0

delayMs(10000);

//for(i=500000; i>0; i--); // Delay

P2->OUT = BIT1; // Toggle P1.0

delayMs(10000);

//for(i=500000; i>0; i--); // Delay

P2->OUT =BIT2; // Toggle P1.0

delayMs(10000);

//for(i=500000; i>0; i--); // Delay

}

}

void delayMs(int n)

{

int i,j;

for(j=0;j<n;j++);

for(i=500000; i>0; i--); // Delay

}

1. Write an embedded c program to perform LED blinking ( P2.0) after pressing i) P1.1, ii) P1.4.

#include "msp.h"

int main(void) {

P1->SEL1 &= ~2; /\* configure P1.1 as simple I/O \*/

P1->SEL0 &= ~2;

P1->DIR &= ~2; /\* P1.1 set as input \*/

P1->REN |= 2; /\* P1.1 pull resistor enabled \*/

P1->OUT |= 2; /\* Pull up/down is selected by P1->OUT \*/

P2->SEL1 &= ~1; /\* configure P2.0 as simple I/O \*/

P2->SEL0 &= ~1;

P2->DIR |= 1; /\* P2.0 set as output pin \*/

while (1) {

if (P1->IN & 2) /\* use switch 1 to control red LED \*/

P2->OUT &= ~1; /\* turn off P2.0 red LED when SW is not pressed \*/

else

P2->OUT |= 1; /\* turn on P2.0 red LED when SW is pressed \*/

}

}

1. Write an embedded c program to generate a delay using SysTickTimer ( 3MHz ) i) of delay 1 sec ii) of delay 2 sec iii) of delay 50 msec.

#include "msp.h"

int main(void){

P2->SEL1 &= ~1;

P2->SEL0 &= ~1;

P2->DIR |= 1;

SysTick->LOAD =3000000-1;

SysTick->VAL=0;

SysTick->CTRL=5;

while(1){

if (SysTick->CTRL & 0x10000)

P2->OUT ^= 1;

}

}

1. Write an embedded c program to generate delay of 2sec using timer32.

# include "msp.h"

void delayMs(int n);

int main(void)

{

uint32\_t i;

P2->DIR = 1;

while(1)

{

P2->OUT |= 1; // SETTING P2.0 AS OUTPUT PORT

delayMs(1);

P2->OUT &= ~1;

delayMs(1);

}

}

void delayMs(int n){

TIMER32\_1->LOAD = n\*6000000;

TIMER32\_1->CONTROL= 0xC2;

while((TIMER32\_1->RIS& 1) == 0);

TIMER32\_1->INTCLR=0;

}

1. Write an embedded c program to square wave of 50% duty cycle.

#include "msp.h"

int main(void) {

/\* initialize P2.0 for red LED \*/

P2->SEL1 &= ~1; /\* configure P2.0 as simple I/O \*/

P2->SEL0 &= ~1;

P2->DIR |= 1; /\* P2.0 set as output \*/

TIMER\_A1->CTL= 0x0211; /\* SMCLK, ID= /1, up mode, TA clear \*/

TIMER\_A1->EX0= 0; /\* Divider2 = 1/(0+1) = 1/1 \*/

TIMER\_A1->CCR[0]=30-1;

while (1) {

while((TIMER\_A1->CCTL[0]& 1) == 0){ /\* wait until the CCIFG is set \*/

TIMER\_A1->CCTL[0]&= ~1; /\* clear interrupt flag \*/

P2->OUT ^= 1; /\* toggle red LED \*/

}

}

}