電子情報工学実験

GCCと最適化

3I14 公文健太

# 実験

3.1 -1

RCALL 3

LDI 1

{

NOP x5 5

SUBI 1

BRNE 2

} x248

NOP x5 5

SUBI 1

BRNE 1

NOP 1

RET 4

すべて足し合わせて、2000クロック

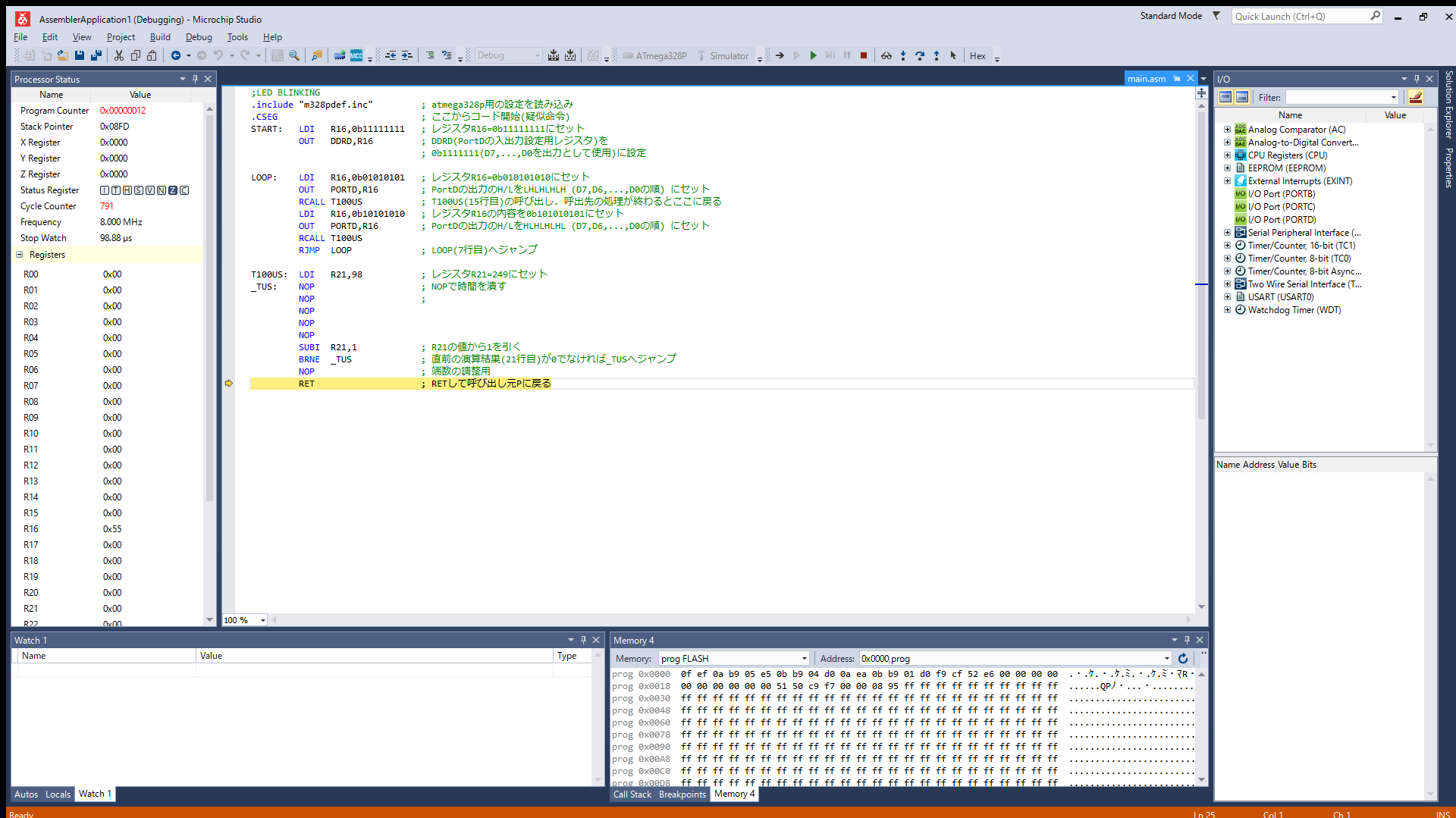
3.1 -2

20 / 8 = 2.5より、必要なクロック数は2000 / 2.5 = 800クロック

3.3 -1のプログラムでは、ループする部分の回数をnとおくとクロック数は16+8nとあらわせるため、16+8n = 800 ⇔ n = 98となるため、

LDI R21, 98に修正すれば100usの遅延になる。

3.1 -3



3.1 -4

構文は OUT A, Rr

オペコードは 1011 1Aar rrrr AAAA

となるので、例のコードでは

OUT 0x04, R24

A: 00 0100

R: 11 0000

のため

1011 1001 1000 0100

16進数に変換すると

b984

入れ替えて

84b9となる。

3.2

one()では

R24 0x01 R25 0x00

twoHandredFiftyEight()では

R24 0x02 R25 0x01

のためR24は下位8bit、R25は上位8bitを表していると考えられる。

3.3 -1

loop.cではdo-while文がsubi, sbc, brneなどに解釈されており、その分unrolledLoop.cにくらべ処理する部分が増えていると考えられる。

3.3 -2

O0のみ遅延ループが削除されなかった。

コンパイルオプションの違いによりファイルサイズが変化した。

ファイルサイズはO0 > O1 = O2 = O3 = Osの順になった。

3.3 -3

ファイルサイズは

O0(5797B) > O3(5629B) > O1(4971B) = O2(4971B) > Os(4939B)

の順になった。

3.3 -4

O0:最適化はなし。デバッグ用に使われる。

O1:最小限の最適化と少しのパフォーマンス向上が見込める。

O2:バランスよく実行速度の向上とコードサイズが削減される。

O3:より最適化を行う。コンパイルの時間は長くなりファイルサイズも増加する。

Os:コードサイズの最適化を行う。実行速度はおろそかになる。

ソース:

<https://gcc.gnu.org/onlinedocs/gcc/Optimize-Options.html#Optimize-Options>

3.3 -5

O0以外のいずれかにすべき

なぜならO0ではfor文が展開されてしまうため無限ループが展開されファイルサイズが大きくなるため。

3.3 -6

5ms

#define F\_CPU 20000000UL

#include <asf.h>

#include <util/delay.h>

int main(void) {

    DDRB = 0x01;

    for(;;){

        \_delay\_ms(5);

        PORTB ^= 0x01;

        \_delay\_ms(5);

        PORTB ^= 0x00;

    }

}

10us

#define F\_CPU 20000000UL

#include <asf.h>

#include <util/delay.h>

int main(void) {

    DDRB = 0x01;

    for(;;){

        \_delay\_us(10);

        PORTB ^= 0x01;

        \_delay\_us(10);

        PORTB ^= 0x00;

    }

}

# 感想

コンパイルオプションによってどのようにコードがアセンブラに変換されるかがわかった。

いままでどのように機械語に変換されているかわからなかったので、lssファイルなどを読んで細かい部分まで理解できたのでよかった。