Tom Almy's Blog

More Dynamic Than His Almy.us Website!

ARDUINO BOARDS / ARDUINO SAMD21-BASED BOARDS

Performance Advantage of ARM over AVR

By Tom on Thursday, February 17, 2022

I've previously shown that the SAMD21 is slower than the ATMega328P when accessing GPIO ports, however it does have a major advantage performing arithmetic. There's quite an advantage to a 32-bit processor over an 8-bit processor.

I was also interested in seeing the performance impact of using floating point, which is done in software on both of these microcontrollers. Integer division is also slow because it, too, is done in software.

First I looked at celsius to Fahrenheit conversion. We will be storing the temperatures in 16 bit integers in tenths of degrees:

```
volatile  int16_t celsius = 370;
volatile  int16_t fahrenheit = 0;
```

The variables are declared volatile to prevent the calculations from being optimized out. We will be converting normal body temperature, 37ºC to, hopefully, 98.6ºF. The statement to do the conversion using integers is:

```
fahrenheit = celsius*9/5 + 320;
```

To be "fair", since the SAMD21 does 32-bit arithmetic, we will also try this statement that forces 32-bit calculations in the ATMega328P:

```
  fahrenheit = celsius*9L/5 + 320;
```

To do the calculation using floating point (32-bit float data type) we will use the following statement. Note that it has to convert celsius to a floating point value and then convert the result back to integer:

```
fahrenheit = int16_t(celsius*1.8f + 320.0f);
```

Executing on an Arduino Uno (ATmega328P) and on an Arduino Nano 33 IoT (SAMD21) we get the following times:

Calculation	**Uno**	**Nano 33 IoT**
Integer	14.8µs	2.8µs
32-bit Integer	40.8µs	2.8µs
32-bit Float	24.8µs	9.4µs

Performance of Temperature Unit Conversion

We see that the SAMD21 is about 5.3 times faster than the ATMega328P for the integer calculation. The penalty for floating point is 3.4x and 1.7x respectively, so floating point is more costly in the SAMD21. Notice that double precision floating point is available for the SAMD21 but isn't used here (I hope!). The big surprise is the major hit in execution time on the Uno if 32-bit integer arithmetic is forced. Floating point ends up being faster!

I then looked at a pair of programs that find prime numbers. I wrote these programs some years ago to test operation of PDP-8 minicomputers being emulated in a class I taught. They find all primes less than 4096, the maximum integer in the PDP-8. The *sieve* program uses a sieve of Eratosthenes approach, building a table of bits of each value, so 4096 bits in total. The program performs no multiplications or divisions. The basic algorithm for filling the table, which will be measured here, is:

```
#define MAXINT 4095     /* Maximum integer */
#define MAXROOT 64      /* We don't need to check beyond this */

#define WORDS ((unsigned)(MAXINT/8))  /
  uint8_t sieve[WORDS];
  unsigned checking;       /* The value we are currently checking */
  unsigned multiple;       /* The location we are marking */
  unsigned count = 0;
  unsigned i, j;           /* Temporary variables */

  /* First clear the array */
  i = 0;
  do
  {
    sieve[i] = 0;
```

```
    i = i + 1;
  } while (i < WORDS);

  /* Do the marking */
  checking = 2;
  do
  { /* Is the value we are checking a prime? */
    i = checking >> 3; /* word index is checking right shifted 3 bits*/
    j = 1 << (checking & 0x7); /* bit index -- shift 1 left 0 to 7 times
                                  depending on 3 lsbs of checking */
    if ((sieve[i] & j) == 0) /* we have a prime, so... */
    {
      /* Mark multiples of the number as not being prime */
      multiple = checking;
      do
      {
        multiple = multiple + checking;
        if (multiple > MAXINT) break;
        /* mark the location as not being prime */
        i = multiple >> 3;      /* again, find the word index */
        j = 1 << (multiple & 0x07); /* and the bit mask */
        sieve[i] = sieve[i] | j; /* Set the bit */
      } while (1);        /* Do forever, until break is executed */
    }
    checking  = checking + 1;       /* Go to next value to check */
  } while (checking < MAXROOT);
```

The second version of the program, *brute*, does a brute force approach and uses the modulo function. The intent was for students that didn't get PDP-8 multiply/divide operating. It also takes much less data memory, which can be important for a microcontroller. Execution time is much longer for this version. Sorry about the goto statement.

```
#define MAXINT 4095     /* Maximum integer with our 12 bit numbers */
#define MAXROOT 64      /* We don't need to check beyond this */

  volatile unsigned checking;       /* The value we are currently checking */
  unsigned trial;          /* The trial divisor */
  unsigned i;
  unsigned count = 0;

  checking = 2;       /* Start with 2 */
  do
  {
    trial = 2;
    while (trial < checking)
```

```
      {
        /* If there is no remainder, then checking is divisible by
           trial */
        if (checking % trial == 0) goto notPrime;
        trial = trial + 1;
      }
      /* If we get here, we have a prime number */
      i = checking;
      // VALUE PRINTED OUT HERE
  notPrime:
      checking = checking + 1; /* Go to next value to check */
    } while (checking <= MAXINT);
```

We see that the SAMD21 is at least 10 times faster.

Metric	Uno	Nano 33 IoT
Sieve time	28.5 ms	2.6 ms
program memory	134 bytes	92 bytes
variable memory	511 bytes	512 bytes
Brute time	14.75 seconds	1.15 seconds
program memory	136 bytes	60 bytes
variable memory	4 bytes	4 bytes

Prime Number Program Performance

Memory sizes are those additional to an empty program. Note that the SAMD21 board uses 11316 bytes of program memory with an empty program while the ATMega328P uses 444 bytes. A lot of the extra use is in library routines that can be used by the application, which reduces the program memory to that shown in the table above. I would expect program memory usage to be roughly the same for large programs.

Compiler selection can also make a difference and the GCC compilers used in the Arduino IDE are known to be good for AVR microcontrollers but not so good for ARM microcontrollers. In my last job before retiring we used the Keil compiler on ARM projects and got surprisingly better code compared to GCC. However it does come at a cost, namely it does cost money and isn't really suitable for hobbyists.

ATmega328P floating point vs integer performance prime number computation SAMD21

Previous Post Next Post

Related Posts

ARDUINO BOARDS / ARDUINO SAMD21-BASED BOARDS

The Circuitous Route To Arduino SAMD Board Program Loading

ARDUINO BOARDS / ARDUINO NANO EVERY / ARDUINO SAMD21-BASED BOARDS

Is the SAMD21 the future for Arduino Boards?

ARDUINO SAMD21-BASED BOARDS / FAR INSIDE THE ARDUINO

The SAMD21 Microcontroller

Recent Posts

The Circuitous Route To Arduino SAMD Board Program Loading

Is the SAMD21 the future for Arduino Boards?

The SAMD21 Microcontroller

Finally Finished 37 Sensor Book

Busy With The New Book

Book Titles

These are available as paperbacks or in Kindle format:

NEW! **Making Sense of 37 Sensors**

Still Far Inside The Arduino

Far Inside The Arduino

Far Inside The Arduino: Nano Every Supplement

The PDP-8 Class Project: Resoling An Old Machine

The following is available only as a paperback book:

Designing with Microcontrollers — The 68HCS12

Archives

July 2022

June 2022

May 2022

April 2022

March 2022

February 2022

December 2021

November 2021

September 2021

August 2021

July 2021

June 2021

May 2021

April 2021

March 2021

February 2021

January 2021

December 2020

November 2020

Search...

About Tom

Now retired, Tom Almy, has written applications for microcontrollers throughout their history, worked in the electronics industry for 43 years and taught courses in digital design and programming for 25 years. He is now looking at electronics from a hobbyist point of view. Tom holds eleven patents, most pertaining to digital and analog circuit design. He received a MSEE degree from Stanford University and a BS with Distinction in Electrical Engineering from Cornell University.

Categories

Arduino Nano Every

Arduino SAMD21-based Boards

Far Inside The Arduino

Meta

PDP-8

Still Far Inside The Arduino

Uncategorized

Recent Posts

The Circuitous Route To Arduino SAMD Board Program Loading

Is the SAMD21 the future for Arduino Boards?

The SAMD21 Microcontroller

Finally Finished 37 Sensor Book

Busy With The New Book

Recent Comments

Search...

Archives

July 2022

June 2022

May 2022

April 2022

March 2022

February 2022

December 2021

November 2021

September 2021

August 2021

July 2021

June 2021

May 2021

April 2021

March 2021

February 2021

January 2021

December 2020

November 2020

October 2020

September 2020

August 2020

July 2020

June 2020

May 2020

March 2020

February 2020

December 2019

November 2019

October 2019

September 2019

August 2019

Meta

Log in

Entries feed

Comments feed

WordPress.org

Meta

Log in

Entries feed

Comments feed

WordPress.org

Pages

About This Site