CS 241 Lab 05
(Arduino Benchmarking)

Benjamin Stream
Solomon Himelbloom

February 15, 2021

1 Answers to Questions

- *How linear is the performance of digitalWrite x1 vs digitalWrite x6?*
- Fairly linear between the x1 and x6 values – there is a slight difference where six single digitalWrite calls are faster than one single x6 digitalWrite call. Otherwise, the respective values follow an almost linear progression as us/call increases.
- 3.23 * 6 = 19.38
- 22.98 / 6 = 3.83
- *How much speed cost is there to using a table of pins? When would a table of pins be worth that cost?*
- There is a slight speed gain of 2.07 us from using a table of pins rather than separate hard-coded calls with the traditional digitalWrite system. A table of pins would be worth the cost when writing loops over non-identical, yet repetitive tasks. Tables are also convenient for efficiently maintaining code, so this cost would outweigh the timing drawbacks.

- *How much speed gain do you get from direct port access? When is that gain worth the cost?*
- From direct port access, we get approximately a 25 percent speed gain from a single digitalWrite call and a 31 percent speed gain from a traditional digitalRead call. This gain is worth the cost when using equipment that requires extremely time-sensitive movements, like that one single servo.
- *How much relative time error (actual delay / requested delay - 1) is there in a 1ms delay vs a 1us delay vs a 10us delay?*
- 1ms = (1008.06 / 1000) - 1 = 0.00806ms
- 1us = (0.76 / 1) - 1 = -0.24us
- 10us = (9.59 / 10) - 1 = -0.041us
- *How could this delay time error affect your program if it needs very precise delays (small relative error)?*
- If we think about how some of these extremely precise delays can be compounded over a long period of time, any added delay could be detrimental tasks that need to be completed in a certain order given an amount of time. When using delays with programs, consider how a small relative error can add up over time and account for this relative error with every new function or input.
- *On Arduino, is it faster to do bitwise operations or addition?*
- Exactly the same time, because both the "AND" as well as the addition operators are executed directly at the CPU level.
- *What's the relative time cost of addition for int vs float?*
- It is roughly a 10 times increase in time, this makes sense because floats are much bigger in the amount of space that they take up.
- *Does the Arduino have divide hardware?*

- Yes, we can tell this due to the fact that the divide operator takes so little time to execute.
- *Does the Arduino have floating point hardware?*
- No, this is evident due to the amazingly slow 8.57 us time compared to the 0.76 us/call operations with hardware at the CPU level.
- *What makes serial communication so slow?*
- Serial communication on our Arduinos is so slow because our device is anticipating a timeout from the code supplied. In all other cases, the Arduino is waiting for the succeeding commands after the delay before executing.

Operation Name	**Details**	**us/call** (Microseconds per operation)
digitalWrite x1	Call digitalWrite for pin 8	3.23
digitalWrite x6	Call digitalWrite for each pin from 8 through 13, using separate hard-coded digitalWrite calls	22.98
digitalWrite x6 Table	Call digitalWrite for each pin from 8 through 13, using a table of pins	20.91
digitalWrite PORT	Use direct port access to do a single hardcoded write operation setting pins 8 through 13	0.82
digitalRead x1	Call digitalRead for pin 8	2.85
digitalRead PORT	Use direct port access to do a digital read.	0.89
analogRead x1	Call analogRead for pin A0	112.01

Operation Name	Details	us/call
delayMicroseconds 1	Ask for a 1 microsecond delay	0.76
delayMicroseconds 10	Ask for a 10 microsecond delay	9.59
delay 1	Ask for a 1 millisecond delay (you can adjust nRepeat to make this benchmark finish sooner)	1008.06

Operation Name	Details	us/call
return 0	Empty function, just return 0	0.76
andInt	Bitwise AND two integers	0.89
addInt	Add two integers	0.89
mulInt	Multiply two integers	1.01
divInt	Divide two integers	0.76
addFloat	Add one float to the int v benchmark parameter	8.57
serialPrint	Print a "(char)" to the serial port (at 9600 baud)	1036.96

2 Appendix

2.1 Source Code

```
#include "benchmark.h" //<- Dr. Lawlor's code

void setup() {
  Serial.begin(9600);
}

// Variables
const int npatterns = 1; // reused code, can be cleaned up
const int npins = 6;
int pattern[npatterns][npins] = {
  {8, 9, 10, 11, 12, 13},
};

// Define functions to be benchmarked.
// These need to take int and return int.
int andInt(int v) {
  int i = 1;
  return v & i;
}

int addInt(int v) {
  int i = 1;
  return v + i;
}

int mulInt(int v) {
  int i = 4;
  return v * i;
}

int divInt(int v) {
  int i = 1;
  return v / i;
}

float addFloat(int v) {
  float i = 1;
  return v + i;
}

int retZero(int v) {
  return 0;
}

int serialPrintFunction(char v) {
  Serial.print(v);
}

int delayMicroS(char v) {
  delayMicroseconds(1);
}

int delayMicroSTen(char v) {
  delayMicroseconds(10);
}

int delaymS(char v) {
  delay(1);
```

```
}

int digWriteOne(char v) {
  digitalWrite(8, HIGH);
}

int digWriteSix(char v) {
  digitalWrite(8, HIGH);
  digitalWrite(9, HIGH);
  digitalWrite(10, HIGH);
  digitalWrite(11, HIGH);
  digitalWrite(12, HIGH);
  digitalWrite(13, HIGH);
}
int digWriteSixHack(char v) {
  PORTB = 0b111111;
}

int digWriteSixArray(int v) {
  for (int i = 0; i < npins; i++) {
    digitalWrite(pattern[i], HIGH);
  }
}

int digReadOne(char v) {
  digitalRead(8);
}

int digReadPort(char v) {
  return PINB;
}

int anaRead(char v) {
  analogRead(A0);
}


void loop() {
  Serial.println("Starting benchmarks...");
  // Call benchmark on our functions
  //  benchmark("addInt",addInt,10000);
  //  benchmark("andInt",andInt,10000);
  //  benchmark("mulInt",mulInt,10000);
  //  benchmark("divInt",divInt,10000);
  //  benchmark("addFloat",addFloat,10000);
  //  benchmark("return 0",retZero,10000);
  //  benchmark("Serial Print",serialPrintFunction,10000);
  //  benchmark("delayMicroseconds", delayMicroS, 10000);
  //  benchmark("delayMicroseconds 10", delayMicroSTen, 10000);
  //  benchmark("delay 1", delaymS, 10000);
  //  benchmark("digitalWrite x1", digWriteOne, 10000);
  //  benchmark("digitalWrite x6", digWriteSix, 10000);
  //  benchmark("digitalWrite x6 HACK (Port)", digWriteSixHack, 10000);
  //  benchmark("digitalWrite x6 Array", digWriteSixArray, 10000);
  //  benchmark("digitalRead Port", digReadPort, 10000);
  //  benchmark("digitalRead One",digReadOne, 10000);
  //  benchmark("analogRead One",anaRead, 10000);
  Serial.println();
  delay(8000);
}
```