School of Electrical and Electronic Engineering

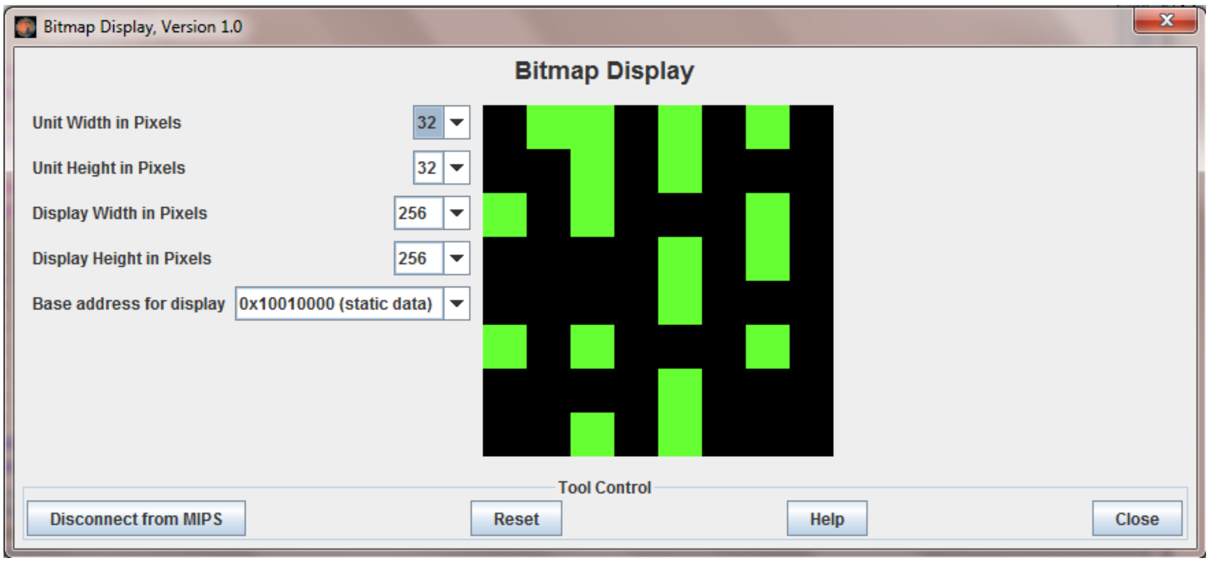
COMP 30080

Processor design

Student Name:	Paul Doherty
Student Number:	10387129
Date:	24/03/2014
Assignment:	Assignment 3

Problem 1

In order to output the prime number between 1-64 on an 8 × 8 bitmap a loop was set up to cycle through the numbers 1-64. An output array to hold the colour that is to be shown on the bitmap was also created. The addresses of this array would be represented by the numbers 1-64 in turn. Each number was then compared to an array consisting of the 18 prime numbers that exist between 1-64. If the numbers matched the corresponding output address is given the RBG colour code that will show on the bitmap. If the number has no match with the prime list then the corresponding address is given the value zero, which represents no colour (black) on the bitmap. The outputted bitmap can be seen in figure 1.



Unit Width in Pixels: 32
Unit Height in Pixels: 32
Display Width in Pixels: 256
Display Height in Pixels: 256
Base address for display: 0x10010000 (static data)

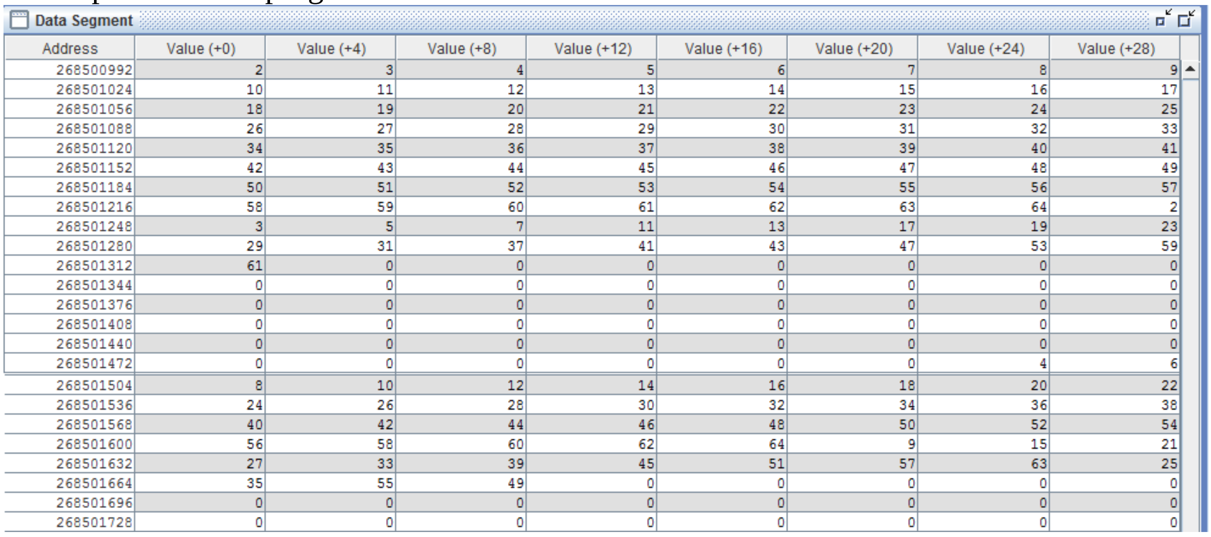
Figure 1: Shows 8X8 grid of prime numbers. Green squares represent prime numbers and black squares represent non-prime integers.

Problem 2

This problem required the algorithmic programming of the Sieve of Eratosthenes method of finding square numbers. To implement this, an array of numbers from 2-64 was created; this is the range of numbers that the algorithm will be applied to. Two output arrays were then created; one to store the detected prime numbers and one store the non-prime numbers.

In the algorithm a prime number is detected by comparing it to the already existing list of non-prime numbers. If the number being tested is not found in the already existing list of non-primes then that number is prime.
After a prime number is found it undergoes two separate operations whose results are stored as non-prime numbers.
The first operation that is performed is the prime number is squared and immediately stored in the list of non-primes. The second operations is then carried out which adds the prime number to is square repeatedly until it exceeds the wanted range of prime numbers which; in this case is 64. After each addition is performed the result is checked against the already existing list of primes, this is to avoid overlapping which occurs frequently in the algorithm. This addition operation is performed until the sum eventually exceeds 64. When this number is exceeded it then tests the next test number against the new list of non-primes and so on. This process is repeated until all numbers 2-64 are tested.

In my implementation of the algorithm, the two output arrays, Prime and NotPrime are both3 words long, while intuitively I know long each needs to be I left them this length for the computers point of view all the numbers could have been prime or not prime. Figure 2 shows the output from this program.



Address	Value (+0)	Value (+4)	Value (+8)	Value (+12)	Value (+16)	Value (+20)	Value (+24)	Value (+28)
268500992	2	3	4	5	6	7	8	9
268501024	10	11	12	13	14	15	16	17
268501056	18	19	20	21	22	23	24	25
268501088	26	27	28	29	30	31	32	33
268501120	34	35	36	37	38	39	40	41
268501152	42	43	44	45	46	47	48	49
268501184	50	51	52	53	54	55	56	57
268501216	58	59	60	61	62	63	64	2
268501248	3	5	7	11	13	17	19	23
268501280	29	31	37	41	43	47	53	59
268501312	61	0	0	0	0	0	0	0
268501344	0	0	0	0	0	0	0	0
268501376	0	0	0	0	0	0	0	0
268501408	0	0	0	0	0	0	0	0
268501440	0	0	0	0	0	0	0	0
268501472	0	0	0	0	0	0	4	6
268501504	8	10	12	14	16	18	20	22
268501536	24	26	28	30	32	34	36	38
268501568	40	42	44	46	48	50	52	54
268501600	56	58	60	62	64	9	15	21
268501632	27	33	39	45	51	57	63	25
268501664	35	55	49	0	0	0	0	0
268501696	0	0	0	0	0	0	0	0
268501728	0	0	0	0	0	0	0	0

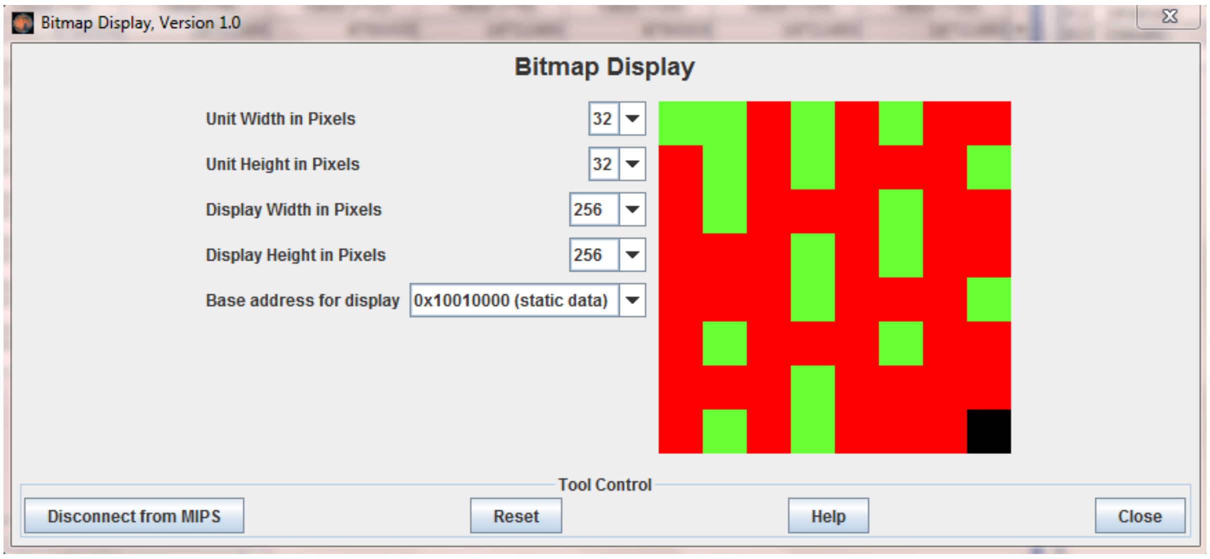
Figure 2: This shows the output of the Sieve of Eratosthenes algorithm. Register addresses 268500992 - 268501240 show the list of integers being tested, addresses 268501244 - 268501492 hold the list of prime numbers found by the algorithm (ignore zeros) and finally registers 268501496-26801672 hold the non-prime numbers detected by the algorithm

Problem 3

This problem required the joining of problems 1 and 2. To implement this I first coloured all squares representing integers 1-64 the same colour. I then used the Sieve of Eratosthenes algorithm find the non-prime numbers. When a non-prime number was detected its corresponding square on the bitmap was coloured red to show it deleted.

This problem involved an extra loop to track the position of the square so that it represented the correct number being deleted, this was implemented by adding a simple counter that incremented each time the address of the bitmap output was updated.

The output can be seen in figure 3.


Bitmap Display, Version 1.0

Bitmap Display

Unit Width in Pixels: 32

Unit Height in Pixels: 32

Display Width in Pixels: 256

Display Height in Pixels: 256

Base address for display: 0x10010000 (static data)

Figure 3: Output of bitmap. The bit map represents numbers 2-64, the black square in the bottom right corner is not being used.