RAMAIAH UNIVERSITY OF APPLIED SCIENCES

ASSIGNMENT

Course Code	CSC208A
Course Name	Computer Organization and Architecture
Programme	B.Tech
Department	CSE
Faculty	FET

Name of the Student	Satyajit Ghana
Reg. No	17ETCS002159
Semester/Year	03/2018
Course Leader/s	Naveeta Rani

Declaration Sheet			
Student Name	Satyajit Ghana		
Reg. No	17ETCS002159		
Programme	B.Tech	Semester/Year	03/2018
Course Code	CSC208A		
Course Title	Computer Organization and Architecture		
Course Date		to	
Course Leader	Naveeta Rani		

Declaration

The assignment submitted herewith is a result of my own investigations and that I have conformed to the guidelines against plagiarism as laid out in the Student Handbook. All sections of the text and results, which have been obtained from other sources, are fully referenced. I understand that cheating and plagiarism constitute a breach of University regulations and will be dealt with accordingly.

Signature of the Student		Date	
Submission date stamp (by Examination & Assessment Section)			

Signature of the Course Leader and date	Signature of the Reviewer and date

Question No. 1

Solution to Question No. 1 Part A:

A 1.1 Introduction to cache memory:

A CPU is meant to perform some basic operations, to do these it needs to store and fetch the values from somewhere, one of the places where it can temporarily store is the registers, they have the lowest latency, but the capacity of these registers is very limited, and the CPU has to store it in Main Memory or RAM, this gives an extra overhead for fetching data, to reduce this time to access data from the main memory, a cache is implemented that is very close to the processor core and stores copies of data that are frequently being accessed from the memory. So when the CPU wants to fetch a data from the Memory it checks if it's there already in the cache, so it will read and write in the cache instead of the main memory. Majority of the CPU's have different cache for difference purposes such as data cache and instruction cache.

Caches are then further divided into hierarchy of L1, L2 and L3 cache. L1 cache is split into L1d(data) to speed up data fetch and store and L1i(instruction) cache to speed up executable instruction fetch. Multi-Core processors have a dedicated L2 cache for each core, and larger processors may even have a L3 or even a L4 cache.

A simple illustration of the cache system. Not to be used for reference

A 1.2 Critical Analysis:

(i) Program Execution Time

The cache performance model can be expressed by the cache factor as a linear combination of execution times that are blended by cache hit and miss rates:

$$f_{cache} \begin{cases} 1 & if\ A_C \\ r_H + r_M \times G & otherwise \end{cases}$$

To quantify the effect of performance on the execution time or the performance the cache model f_{cache} is proposed and is examined via

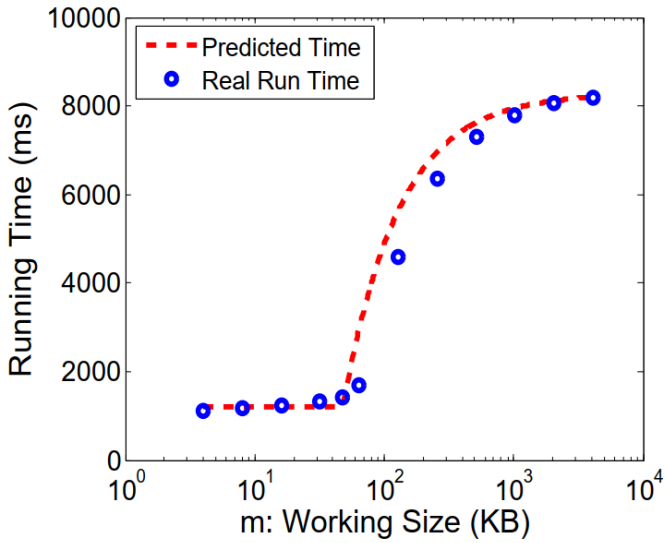

Figure A1.1 Impact of cache on execution time

the micro-benchmark. Where r_H is the cache hit rate and r_M is the cache miss rate, the benchmark is carries out for various sizes of cache and the predicted model and the actual results are plotted in Figure A1.1. Dramatic increase in cache misses and decreases in cache hits are observed once a larger-than-cache working set size m is used.

(ii) Variation in processor speed and cache memory size

- Low Clock Cycles per Instruction (CPI) machines suffer from relative to some fixed CPI memory penalty
    - A machine with a CPI of 5 suffers little from a 1 CPI penalty
    - However, a processor with a CPI of 0.5 has its execution time tripled!
- Cache miss penalties are measures in nanoseconds
    - This means that a faster machine will stall more cycles on the same memory systems.
- Amdahl's law raises again
    - Fast machines with a low CPI are affected significantly from memory access penalties.

The basic key to increasing the computer processing speed is to allow a faster flow of data from the processor and the memory, the faster the processor is the more is the cache required to reduce the *waiting time* for information. It enables the processor to work at its peak performance.

A 1.3 The stance taken with justification:

If we take into consideration only the cache size, it's not enough to compare two CPU's performance relatively. Slower processor with a large cache may be helpful in cases when the cache is the bottleneck for the CPU, larger cache can also abruptly decrease the performance since the rate of cache miss would increase. Faster Processors with smaller caches are definitely a bottleneck since the cache is limiting the processor from achieving its peak performance.

Hence I do not completely agree to the statement, the answer is that it depends, there's a sweet spot of ratio of Clock Speed and Cache memory, as seen from the Figure A1.1, there is a drastic increase in performance after a certain level of cache memory is available for the CPU, but the graph flattens out and starts to decrease after reaching the peak efficiency.

Question No. 2

Solution to Question No. 1 Part B:

Given Data:

X : 5

Y : 9

B 1.1 Explain the steps to divide X by Y using restoring and non-restoring division algorithms:
In our problem 5 should be divided with 9.

$Dividend = 5, Divisor = 9$

Restoring Division:

Here we take Q as the quotient, A as the remainder. The $n - bit$ dividend is loaded in Q and the divisor is loaded in M.

n	M	A	Q	$Operation$
4	01001	00000	0101	Initialize
	01001	00000	101_	Left Shift AQ
	01001	10111	101_	A = A - M
	01001	00000	1010	Q[0]=0 and restore A
3	01001	00001	010_	Left Shift AQ
	01001	11000	010_	A = A - M
	01001	00001	0100	Q[0]=0 and restore A
2	01001	00010	100_	Left Shift AQ
	01001	11001	100_	A = A - M
	01001	00010	1000	Q[0]=0 and restore A
1	01001	00101	000_	Left Shift AQ
	01001	11100	000_	A = A - M
	01001	00101	0000	Q[0]=0 and restore A

Hence the obtained answer is $\boldsymbol{M} - Divisor = 9, \boldsymbol{A} - Remainder = 5, \boldsymbol{Q} - Quotient = 0$

$$5 = 9 \times 0 + 5$$

Steps to perform this form of division:

1. Initialize the content of M with divisor, A with 0 and Q with Dividend, the number of bits is n
2. AQ is treated as a single unit and left shifted
3. The new content of A is A – M or the sum of A and 2's complement of M

4. If the most significant bit of A is 0 then Q is padded with 1, else if the MSB of A is 1 then Q is padded with 0.
5. The value of n is decremented by 1
6. If the value of n is not 0 then repeat from Step 2
7. Now, Q contains the Quotient and A contains the remainder

<u>Non-Restoring Division:</u>

Here we take Q as the quotient, A as the remainder. The $n - bit$ dividend is loaded in Q and the divisor is loaded in M.

n	M	A	Q	$Operation$
4	01001	00000	0101	Initialize
	01001	00000	101_	Left Shift AQ
	01001	10111	101_	A = A - M
3	01001	10111	1010	Q[0]=0
	01001	01111	010_	Left Shift AQ
	01001	11000	010_	A = A + M
2	01001	11000	0100	Q[0]=0
	01001	10000	100_	Left Shift AQ
	01001	11001	100_	A = A + M
1	01001	11001	1000	Q[0]=0
	01001	10011	000_	Left Shift AQ
	01001	11100	000_	A = A + M
0	01001	11100	0000	Q[0]=0
	01001	00101	0000	A = A + M

Hence the obtained answer is $\boldsymbol{M} - Divisor = 9, \boldsymbol{A} - \ Remainder = 5, \boldsymbol{Q} - Quotient = 0$

$$5 = 9 \times 0 + 5$$

Steps to perform this form of division:

1. Initialize the content of M with divisor, A with 0 and Q with Dividend, the number of bits is n
2. If the Sign Bit or the MSB of A is 1 then Left Shift AQ and replace A with sum of A and M
3. Else if the Sign Bit or the MSB of A is 0 then Left Shift AQ and replace A with sum of A and 2's complement of M.
4. If the most significant bit of A is 0 then Q is padded with 1, else if the MSB of A is 1 then Q is padded with 0.

5. The value of n is decremented by 1
6. If the value of n is not 0 then repeat from Step 2
7. If the sign bit of A is 1 then replace A with sum of A and M
8. Now, Q contains the Quotient and A contains the remainder

B 1.2 Select and justify the better option in terms of cost:

Let's analyze the number of cycles for the operations performed in these two division techniques, both these techniques are slow methods for division that make use of the partial remainder concept.

In restoring division if we calculate the division for $n - bits$ and we need as many as $2 \times n$ cycles to land on the final answer. This is the worst case, since there are n cycles for the trial subtractions and another at-max n cycles for the restoration that takes place. This issue is eliminated by a better form of division which is non-restoring division method.

In non-restoring division if we calculate the division for $n - bits$ we need a maximum of $n + 1$ cycles compute the final answer. This is the worst case for Non-Restoring Division. The n cycles are the operations $A + M$ or $A - M$ depending upon the MSB of AQ.

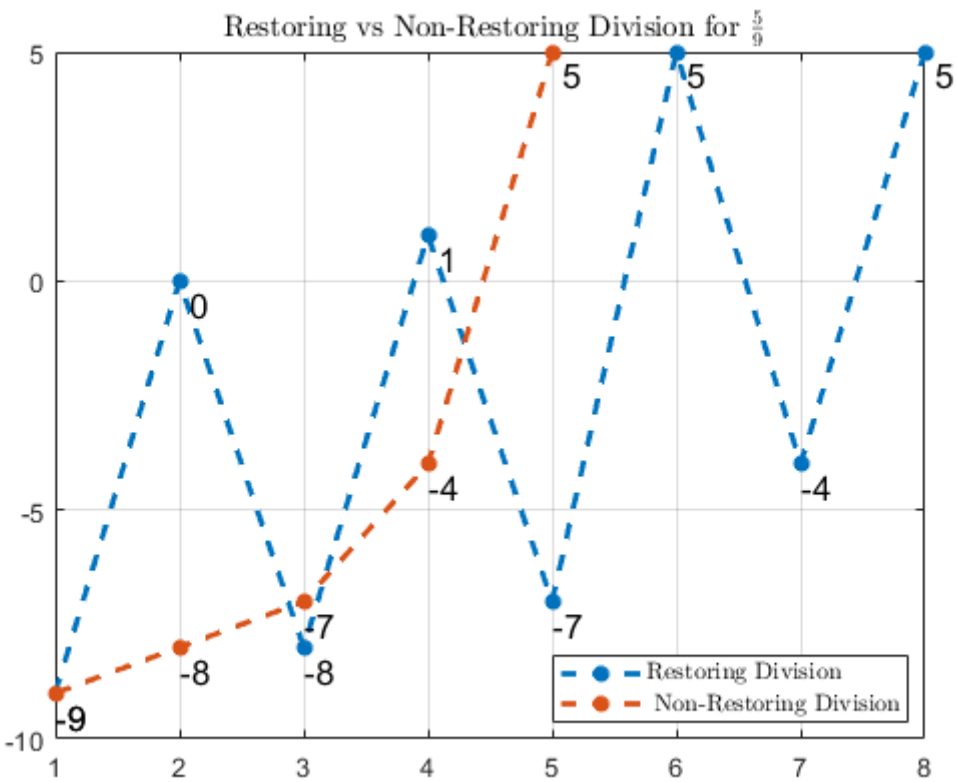

Figure B1.1 Restoring vs Non-Restoring Division

matlab-code for the plot:

```
res_div = [-9 0 -8 1 -7 5 -4 5]
nonres_div = [-9 -8 -7 -4 5]
```

```
plot(1:1:length(res_div), res_div, '--*', 'LineWidth', 2);
hold on;
plot(1:1:length(nonres_div), nonres_div,'--*', 'LineWidth', 2);
grid on;
legend({'$ $Restoring Division', '$ $ Non-Restoring
Division'},'Interpreter', 'latex', 'Location', 'best')
rt = cellstr(num2str(res_div'))
nrt = cellstr(num2str(nonres_div'))
dx = 0.9; dy = -0.5;
text(1:1:length(res_div)+dx, res_div + dy, rt, 'FontSize', 13);
text(1:1:length(nonres_div)+dx, nonres_div + dy, nrt, 'FontSize', 13);

title({'$ $ Restoring vs Non-Restoring Division for $\frac{5}{9}$'},
'Interpreter', 'latex');
```

It' clearly visible that the cost for Non-Restoring division is less in terms of modification of the A Register, assuming changing values is the Rate Determining Step, i.e. the most time consuming, Non-Restoring division makes $n + 1$ changes at max and Restoring Division makes $2n + 1$ changes. Hence Non-Restoring division is better approach from the two.

B 1.3 Conclusion:

Either it is Restoring Division or Non-Restoring Division, both are very tedious and non-optimized forms of division algorithm, other algorithms such as Newton-Raphson Division and Goldschmidt Division are far more efficient and faster algorithms to compute the division. But why do I say that RD and NRD are bad? one is that they are really cumbersome, moreover when division is done for floating points number it gets very-very complex and the efficiency drastically decreases.

Division algorithms are very important in the field of Cryptography where integers with thousands and millions of decimal digits are divided in modular reductions. Algorithms such as Barrett reduction and Montgomery reduction are used in such situations.

Question No. 3

Solution to Question No. 2 Part B:

Given Data:

$X\%$: 50

$Y\%$: 90

Instruction	LOAD	FPADD	FPMUL	Others
Frequency (%)	20	20	10	50
Execution Time (nanoseconds)	90	210	420	200

B 2.1 Introduction to MIPS Performance measure:

The speed of a given CPU depends on many factors, such as the type of instructions being executed, the execution order and the presence of branch instructions (problematic in CPU pipelines). CPU instruction rates are different from clock frequencies, usually reported in Hz, as each instruction may require several clock cycles to complete or the processor may be capable of executing multiple independent instructions simultaneously. MIPS can be useful when comparing performance between processors made with similar architecture (e.g. Microchip branded microcontrollers), but they are difficult to compare between differing CPU architectures. **(Ted MacNeil)**

B 2.2 Compute the MIPS rating of the processor P:

Let's assume a total of 1 Instruction being executed, and the Clock Rate to be F

$$CPI = \frac{Execution\ Time \times Clock\ Rate}{Instruction\ Count}$$

$$CPI = Execution\ Time \times Clock\ Rate$$

$$Average\ CPI = CPI_1 + CPI_2 + CPI_3 + CPI_4$$

$$Average\ CPI = 0.2 \times (90ns \times F) + 0.2 \times (210ns \times F) + 0.1 \times (420ns \times F) + 0.5 \times (200ns \times F)$$

$$Average\ CPI = 202 \times 10^{-9} \times F$$

$$MIPS\ for\ Processor\ P = \frac{F}{CPI \times 10^6}$$

$$MIPS\ for\ Processor\ P = \frac{F}{202 \times 10^{-9} \times 10^6 \times F}$$

$$= 4.95049505\ MIPS$$

B 2.3 Compute the MIPS rating if fast memory chips are used:
If faster memory chips are used then the LOAD Instruction is reduced by 50% and others by 90%, the new execution times are:

Instruction	LOAD	FPADD	FPMUL	Others
Frequency (%)	20	20	10	50
Execution Time (nanoseconds)	$90 \times 0.5 = 45$	$210 \times 0.1 = 21$	$420 \times 0.1 = 42$	$200 \times 0.1 = 20$

Let's assume a total of 1 Instruction being executed, and the Clock Rate to be F

$$CPI = \frac{Execution\ Time \times Clock\ Rate}{Instruction\ Count}$$
$$CPI = Execution\ Time \times Clock\ Rate$$

$$Average\ CPI = CPI_1 + CPI_2 + CPI_3 + CPI_4$$
$$Average\ CPI = 0.2 \times (45ns \times F) + 0.2 \times (21ns \times F) + 0.1 \times (42ns \times F) + 0.5 \times (20ns \times F)$$
$$Average\ CPI = 27.4 \times 10^{-9} \times F$$

$$MIPS\ for\ Processor\ P = \frac{F}{CPI \times 10^6}$$
$$MIPS\ for\ Processor\ P = \frac{F}{27.4 \times 10^{-9} \times 10^6 \times F}$$

$$= 36.49635036\ MIPS$$

B 2.4 Compute the MIPS rating if floating point co-processor is used:
If a floating point coprocessor is used then the execution time of floating point operations (FPAPP and FPMUL) are reduced 3 times or to 33.33%, hence now the execution times are:

Instruction	LOAD	FPADD	FPMUL	Others
Frequency (%)	20	20	10	50
Execution Time (nanoseconds)	90	$\frac{210}{3} = 70$	$\frac{420}{3} = 140$	200

Let's assume a total of 1 Instruction being executed, and the Clock Rate to be F

$$CPI = \frac{Execution\ Time \times Clock\ Rate}{Instruction\ Count}$$
$$CPI = Execution\ Time \times Clock\ Rate$$

$$Average\ CPI = CPI_1 + CPI_2 + CPI_3 + CPI_4$$
$$Average\ CPI = 0.2 \times (90ns \times F) + 0.2 \times (70ns \times F) + 0.1 \times (140ns \times F) + 0.5 \times (200ns \times F)$$

$Average\ CPI = 146 \times 10^{-9} \times F$

$$MIPS\ for\ Processor\ P = \frac{F}{CPI \times 10^6}$$

$$MIPS\ for\ Processor\ P = \frac{F}{146 \times 10^{-9} \times 10^6 \times F}$$

$$= 6.849315068\ MIPS$$

B 2.5 Based on observations B2.3 and B2.4, select and justify the better option:

From the MIPS rating of using faster chips and a floating point co-processor, it's clearly visible that a MIPS score of 36.50 is much greater than 6.85, hence faster chips in this case are a better alternative to increase the performance for Processor P.

1. Ted MacNeil (2004), *Don't be Misled by MIPS*, http://ibmsystemsmag.com/mainframe/tipstechniques/systemsmanagement/don-t-be-misled-by-mips/
2. Lin Ma (2014), *Modeling Algorithm Performance on Highly-threaded Many-Core Architectures*.