Experiment No 1

Design of Ripple Carry Adders

Title: Write a program to implement ripple carry adder.

Objective: To understand the operation of a ripple carry adder, specifically how the carry ripples through the adder.

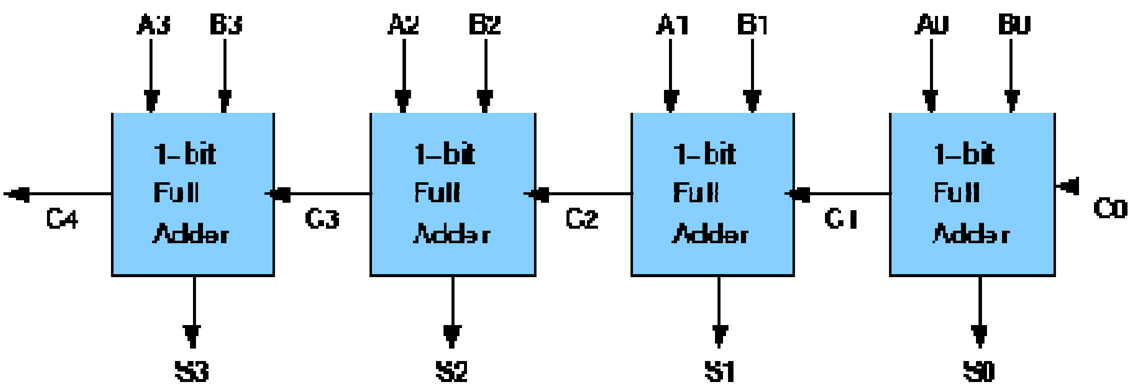
Pre-requisites:
1. Binary Arithmetic
2. Ripple Carry Adder Theory

References:

1. Carl Hamacher, Zvonko Vranesic and Safwat Zaky, "Computer Organization", Fifth Edition, McGraw-Hill.
2. John P. Hayes, "Computer Architecture and Organization", Third Edition.
3. William Stallings, "Computer Organization and Architecture: Designing for Performance", Eighth Edition, Pearson.

Theory:

The block diagram of 4-bit Ripple Carry Adder is shown here below -



The layout of ripple carry adder is simple, which allows for fast design time; however, the ripple carry adder is relatively slow, since each full adder must wait for the carry bit to be calculated from the previous full adder. The gate delay can easily be calculated by inspection of the full adder circuit. Each full adder requires three levels of logic. In a 32-bit [ripple carry] adder, there are 32 full adders, so the critical path (worst case) delay is 31 * 2(for carry propagation) + 3(for sum) = 65 gate delays.

The corresponding boolean expressions are given here to construct a ripple carry adder. In the half adder circuit the sum and carry bits are defined as

sum = A ⊕ B

carry = AB

In the full adder circuit the Sum and Carry output is defined by inputs A, B and Carry in as

Sum =ABC + ABC + ABC + ABC

Carry=ABC + ABC + ABC + ABC

Having these we could design the circuit. But we first check to see if there are any logical equations that would lead to a more structured equation circuit.

With a little algebraic manipulation, one can see that

Sum = ABC + ABC + ABC + ABC

=(AB + AB)C +(AB + AB)C

=(A ⊕ B)C +(A ⊕ B)C

=A ⊕ B ⊕ C

Carry= ABC + ABC + ABC + ABC

= AB +(AB + AB)C

= AB +(A ⊕ B)C

Conclusions.

Post Lab Assignments.

- What is the gate delay in 32 bit ripple carry adder?
- What is the disadvantage of ripple-carry adder?
- What is the major difference between half-adders and full-adders?

Expem ent No 2

Design of Carry Lo Ahead Adders

itle: r pro g o pl en C rr oo h er.

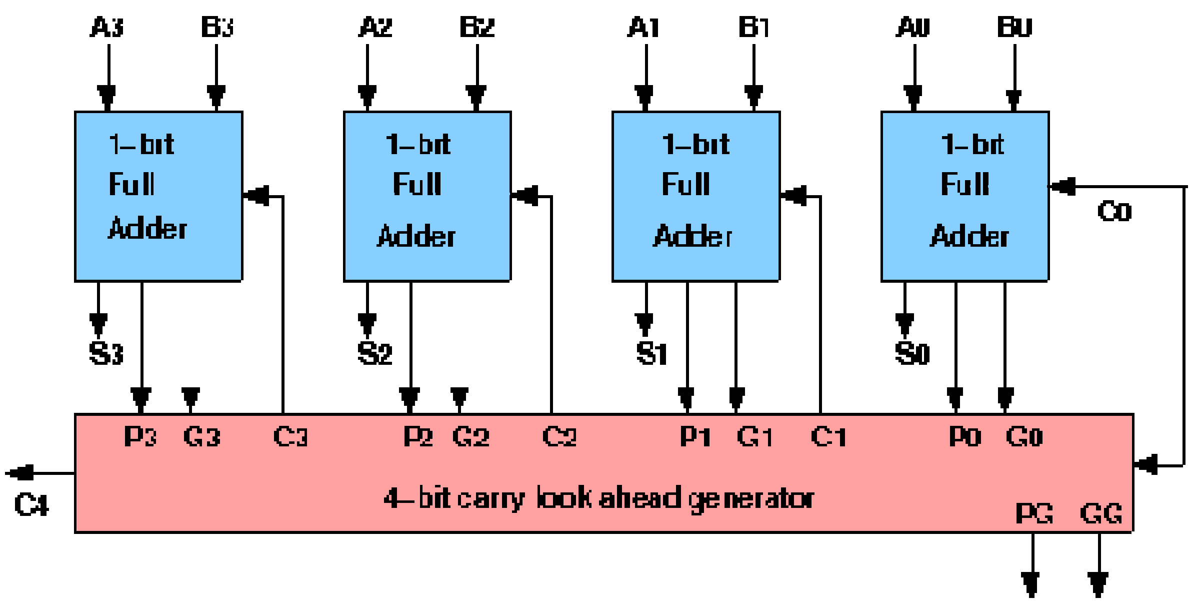
Ob ective: o un er n h oper on o C rr oo h er,s pec c ll h w co pu h c rr p r ll l hus g l s pe n gup h co pu on.

Pre-requisites: 1. B n r Ar h e c
2. C rr oo Ah A er h or

References:
1. C rl h c er, Z on o r ne c n S w Z , "Co pu er r gn on", F h E on, M c r w - ill.
2. o n . . . , "Co pu er Arch ec ure n r gn on", h r E on.
3. ll S ll n g, "Co pu er r gn on n Arch ec ure: De gn n g or er or nce", E gh E on, e s on.

heory:

o re uce h co pu on e, here re s e w s o w o b n r nu bers b us n g c rr loo h ers. h s w or b cre n g w o s gn ls n e n o be C rr rop g or n C rr ener or. h **carry propagator** s prop g e o h nex le el, here s h **carry generator** s us o gener e h ou pu c rr , re g r less o npu c rr . h bloc g o 4-b C rr oo h A er s s ho n h ere belo



h nu ber o g e le els or h c rr prop g on c n be oun ro h c rcu o ull er. h s gn l ro npu c rr C n o ou pu c rr Cou requ res n A D g e n n g e, w h ch

constitute two gate delays. Since there are four full adders in the parallel adder, the output carry C_5 would have 2 X 4 = 8 gate delays from C_1 to C_5. For an n-bit parallel adder, there are 2n gate delays to propagate through.

Design Issues :

The corresponding boolean expressions are given here to construct a carry look ahead adder. In the carry-look ahead circuit we need to generate the two signals carry propagator (P) and carry generator (G),

$$P_i = A_i \oplus B_i$$

$$G_i = A_i \cdot B_i$$

The output sum and carry can be expressed as

$$Sum_i = P_i \oplus C_i$$

$$C_{i+1} = G_i + (P_i \cdot C_i)$$

Having these we could design the circuit. We can now write the Boolean function for the carry output of each stage and substitute for each C_i its value from the previous equations:

$$C_1 = G_0 + P_0 \cdot C_0$$

$$C_2 = G_1 + P_1 \cdot C_1 = G_1 + P_1 \cdot G_0 + P_1 \cdot P_0 \cdot C_0$$

$$C_3 = G_2 + P_2 \cdot C_2 = G_2 P_2 \cdot G_1 + P_2 \cdot P_1 \cdot G_0 + P_2 \cdot P_1 \cdot P_0 \cdot C_0$$

$$C_4 = G_3 + P_3 \cdot C_3 = G_3 P_3 \cdot G_2 P_3 \cdot P_2 \cdot G_1 + P_3 \cdot P_2 \cdot P_1 \cdot G_0 + P_3 \cdot P_2 \cdot P_1 \cdot P_0 \cdot C_0$$

Conclusions.

Post Lab Assignments.

- What is the worst case delay of an n-bit carry look ahead adder?
- What is the advantage of carry look ahead adder over ripple carry adder?
- Design 16 bit CLA using four 4 bit CLA?

Experiment No

Unsigned binary multiplication

Aim: Write a program for unsigned binary numbers multiplication.

Objective: To understand the working of unsigned binary numbers multiplication.

Pre-requisites:

1. Binary Arithmetic
2. Application of Array Adder

References:

1. Carl Hamacher, Zvonko Vranesic and Safwat Zaky, "Computer Organization", Fifth Edition, McGraw-Hill.
2. John P. Hayes, "Computer Architecture and Organization", Third Edition.
3. William Stallings, "Computer Organization and Architecture: Designing for Performance", Eighth Edition, Pearson.

Theory:

Binary numbers are closely related to digital electronics. With digital electronics a '1' means current/electricity is present and a '0' means it's not present. This makes it easier for computers to count through pulses of current (1s and 0s). As we all know, computers can calculate complex equations and perform complex tasks at lightning speed. Calculating using only 1s and 0s is called the BINARY SYSTEM.

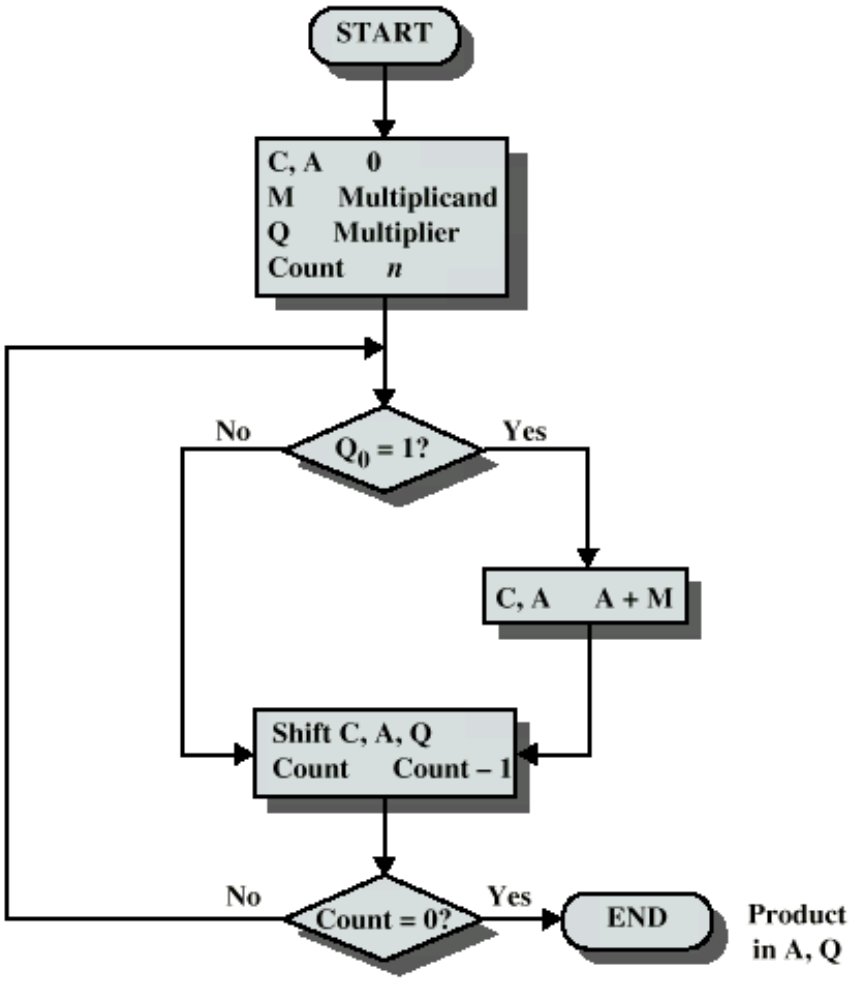
Pen-and-pencil multiplication is similar to the multiplication performed by paper and pencil. This method adds the multiplicand X to itself Y times, where X denotes the multiplicand and Y denotes multiplier. To multiply two numbers by paper and pencil, the algorithm is to take the digits of the multiplier one at a time from right to left, multiplying the multiplicand by a single digit of the multiplier and placing the intermediate product in the appropriate positions to the left of the earlier results.

In unsigned binary multiplication algorithm, A stores the most significant bits of the result, C is the carry bit for **A**, **Q** initially stores the multiplier but then stores the less significant bits of the result. **C, A, Q** are together used as whole final result available in **A** and **Q**.

Execution of Example

C	A	Q	M		
0	0000	1101	1011	Initial Values	
0	1011	1101	1011	Add	First Cycle
0	0101	1110	1011	Shift	
0	0010	1111	1011	Shift	Second Cycle
0	1101	1111	1011	Add	Third Cycle
0	0110	1111	1011	Shift	
1	0001	1111	1011	Add	Fourth Cycle
0	1000	1111	1011	Shift	

Flowchart for Unsigned Binary Multiplication



Conclusion:

Post Lab Questions:

1. Show the multiplication of 12 and 14 using unsigned binary multiplication algorithm.
2. Explain booth's algorithm.
3. Write down the description using binary multiplication algorithm.

Experiment No 4

Booth's Algorithm

Title: Write a program to implement Booth's algorithm for multiplication.

Objective: i) To understand the working of Booth's multiplication algorithm.

ii) To understand the advantages of Booth's multiplier

- It can handle signed integers in 2's complement notation
- It decreases the number of addition and subtraction
- It requires less hardware than combinational multiplier
- It is faster than straightforward sequential multiplier

Pre-requisites: 1. Binary Arithmetic

2. Booth's Algorithm Theory

References:

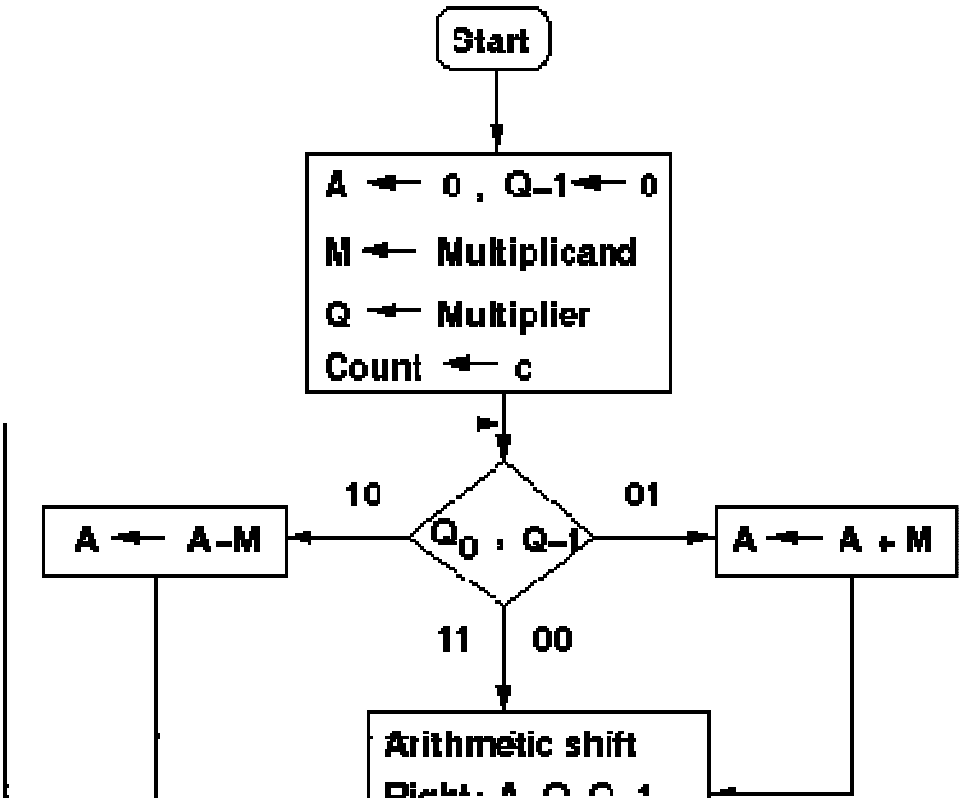
1. Carl Hamacher, Zvonko Vranesic and Safwat Zaky, "Computer Organization", Fifth Edition, Tata McGraw-Hill.
2. William Stallings, "Computer Organization and Architecture: Designing for Performance", Eighth Edition, Pearson.

Theory: Booth's algorithm is depicted as shown in figure below and can be described as follows:

The multiplicand and multiplier are placed in the M and Q registers respectively. A 1 bit register is placed logically to the right of the LSB (least significant bit) Q_0 of Q register. This is denoted by Q_{-1}.

A and Q_{-1} are initially set to 0. Control logic checks the two bits Q_0 and Q_{-1}. If the two bits are same (00 or 11) then all of the bits of A, Q, Q_{-1} are shifted 1 bit to the right. If they are not the same and if the combination is 10 then the multiplicand is subtracted from A and if the combination is 01 then the multiplicand is added with A.

In both the cases results are stored in A, and after the addition or subtraction operation, A, Q, Q_{-1} are right shifted. The shifting is the arithmetic right shift operation where the left most bit namely, A_{n-1} is not only shifted into A_{n-2} but also remains in A_{n-1}. This is to preserve the sign of the number in A and Q. The result of the multiplication will appear in the A and Q.


Start
A ← 0 , Q–1 ← 0
M ← Multiplicand
Q ← Multiplier
Count ← c
10
01
A ← A–M
Q_0 , Q–1
A ← A + M
11
00
Arithmetic shift

Experiment No 5

Restoring and Non –Restoring Algorithm

Title: Write a program to implement restoring and non –restoring algorithm for Division.

Objective:

1. Understanding behaviour of division algorithm for unsigned numbers
2. Implementing restoring and non –restoring algorithms

References:

1. Computer Organization – Carl Hamacher, Zvonko Vranesic, Safwat Zaky
2. Computer Organization and Architecture – William Stallings

Theory:

Division operation is implemented as follows: Position the divisor appropriately with respect to the dividend and performs subtraction. If the remainder is zero or positive, a quotient bit of 1 is determined, the remainder is extended by another bit of the dividend, the divisor is repositioned, and another subtraction is performed. On the other hand, if the remainder is negative, a quotient bit of 0 is determined, the dividend is restored by adding back the divisor, and the divisor is repositioned for another subtraction.

Restoring Division

An n-bit positive divisor is loaded into register M and an n-bit positive dividend is loaded into register Q at the start of the operation. Register A is set to 0. After the division is complete, the n-bit quotient is in register Q and the remainder is in register A. The required subtractions are facilitated by using 2's complement arithmetic. The extra bit position at the left end of both A and M accommodates the sign bit during subtractions.

Nonrestoring Division

The restoring division algorithm can be improved by avoiding the need for restoring A after an unsuccessful subtraction. Subtraction is said to be unsuccessful if the result is negative. Consider the sequence of operations that takes place after the subtraction operation in the preceding algorithm. If A is positive, we shift left and subtract M, that is, we perform 2A – M. If A is negative, we restore it by performing A + M, and then we shift it left and subtract M. This is equivalent to performing 2A + M. The q0 bit is appropriately set to 0 or 1 after the correct operation has been performed.

Algorithm: **Restoring Division**

Do the following n times:

1. Shift A and Q left one binary position.

2. Subtract M from A, and place the answer back in A.
3. If the sign of A is 1, set q_0 to 0 and add M back to A (that is, restore A); otherwise, set q_0 to 1.

Nonrestoring Division

Step 1: Do the following n times:

1. If the sign of A is 0, shift A and Q left one bit position and subtract M from A; otherwise, shift A and Q left and add M to A.
2. Now, if the sign of A is 0, set q_0 to 1; otherwise, set q_0 to 0.

Step 2: If the sign of A is 1, add M to A

Conclusion:

Post Lab Assignment:

1. Using any two methods, perform the operation A/B on the 5-bit unsigned numbers A = 10101 and B = 00101
2. Explain how to perform division on signed operands.

Experiment No 6

ALU Design

Title: ... pro... ... design ... ... logic unit

Objective: ... understanding ... your own ... logic unit
Designing ... logic unit ... given parameters

References.

1. C... Hamacher, Z... Vranesic and S... Zaky, "Computer Organization", Fifth Edition, McGraw-Hill.
2. John P. Hayes, "Computer Architecture and Organization", Third Edition.
3. William Stallings, "Computer Organization and Architecture: Designing for Performance", Eighth Edition, Pearson.

Theory:

A... or Arithmetic Logic Unit ... digital circuit ... to perform arithmetic operations like addition, subtraction, division, multiplication and logical operations like and, or, xor, nand, nor etc.
A simple block diagram of a 4 bit ALU for operations and, or, xor and Add is shown here:

Design Issues:

The circuit functionality of a 4 bit ALU is shown here, depending upon the control signal S1 and S0 the circuit operates as follows:

For Control signal S1 = 0, S0 = 0, the output is **A And B**,

For Control signal S1 = 0, S0 = 1, the output is **A Or B**,

For Control signal S1 = 1, S0 = 0, the output is **A Xor B**,

For Control signal S1 = 1, S0 = 1, the output is **A Add B**.

The truth table for 16-bit ALU with capabilities similar to 74181 is shown here:

Required functionality of ALU (inputs and outputs are active high)

Mode Select Inputs				F_n for active HIGH operands: Logic (M = H)	Arithmetic (note 2) (M = L) (C_n=L)
S3	S2	S1	S0		
				A'	A
			H	A'+B'	A+B
		H		A'B	A+B'
		H	H	Logic 0	minus 1
	H			(AB)'	A plus AB'
	H		H	B'	(A + B) plus AB'
	H	H		A ⊕ B	A minus B minus 1

	1	1	1	AB'	AB minus 1
1				A'+B	A plus AB
1			1	(A ⊕ B)'	A plus B
1		1		B	(A + B') plus AB
1		1	1	AB	AB minus 1
1	1			logic 1	A plus A (to 1)
1	1		1	A+B'	(A + B) plus A
1	1	1		A+B	(A + B') plus A
1	1	1	1	A	A minus 1

The denotes the logic low and 1 denotes logic high.

Conclusions:

Post Lab Assignments:

- Describe with the schematic diagram, how the ALU is connected to the processor registers through common bus.
- Design 4-bit ALU with capabilities similar to 74181.

Expe ent No

Direct Mapping

itle. r pro g o pl n D rec pp n g

Objective: 1. n n n g our o rec pp c c
2. D g n g rec pp c c or g n p r

References. 1. Co pu r r g n on – C ch r, Z on r n c , w Z
2. Co pu r r g n on n Arch c ur – S n g

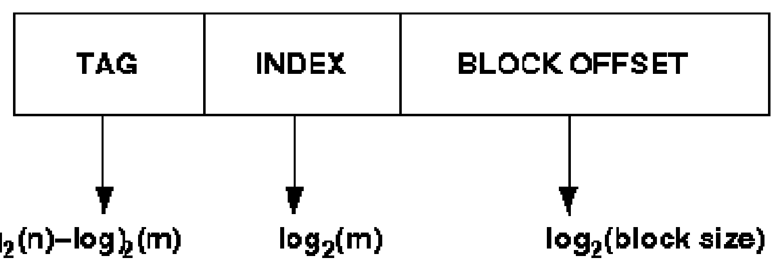
heory. *Direct mapping:*

A given e ory b oc can be apped into one and on y cache ine.

B oc identification. c n or con n bloc s r qu r o g(n)) n c con bloc s , s o n/ ren bloc s o or c n b pp (ren) o c bloc . E c c bloc s g n bloc o or s curr n pr n n c bloc s o con n b o ur r or bloc s n c bloc curr n .

- **Nu ber of bits in the tag: (og n)**
- **Nu ber of sets in the Cache:**
- **Nu ber of bits to identify the correct set: (og)**

or r s n o 3 p rs - g os SB), n x, bloc o s (os SB) n or r o o c pp n g



s s s pl chn qu ch ps ch bloc o n or n o onl on poss bl c n . pp n gs xpr s s

= j o ulo

h r

= c n nu b r

j = n or bloc nu b r

= nu b r o n n c

Algorith :

1. Enter o l nu ber o bloc s n n e or =
2. Enter o l nu ber o c c e e or bloc s = b
3. F n e n e or bloc nu ber =
1. C lcul e e gb o c c e e or b c = /b
2. C lcul e e nu ber o c c e bloc b = o ulo b
3. u pu gb n c c e bloc nu ber
4. S op.

Conclusion:

Post Lab Assign ents :

- A rec pp e c c e cons s s o l (bloc s n e or con ns l (bloc s o 8 b es e c . s e n e or ress or . e s e o e c e ?
- For rec pp e c c e e g 32-b ress, e ollo n gb s o ress re use o ccess e c c e. (Wor = 4 b es): g=31-14, n ex=13-1, o se =5-0. W er e ollo n g
  1. s e c c e bloc s e (Wor s)?
  2. s e s e o e c c e?
- re e n ges o us n g rec pp c c e?

Expe ent No

et Associative Mapping

e: r ro g o pl n s ss oc pp n g

Ob ective: 1). n s n n g bl our o ss oc c ro w or n g o ul

2).D gn n g ss oc c or g n p r

References:

1. C l ch r, Z on r n c n S w Z , "Co pu r r gn on", F E on, c r w - ll.
2. ohn . , "Co pu r Arc c ur n r gn on", r E on.
3. ll S ll n g, "Co pu r r gn on n Arc c ur : D gn n g or r or nc ", E g E on, s on.

heory: Co b n on o r c n ss oc pp n g c n qu o g r or s s ss oc pp n g r bloc s o c r group n os s n pp n g ll w s bloc o n or o r n n bloc o s p c c s nc con n on probl o r c pp n gs s b n g w c o c or bloc pl c n A s w r cos s r uc b c s n g s o ss oc s rc .

nu b r o bloc s p rs s p r r c n b s l c os u r qu r n o p r cul r co pu r. A c c s - bloc s p rs s r rr o s w s ss oc c c n s pp n g c n qu s c ll w s ss oc pp n g

For s ss oc pp n g ollw n g or ul s us o n p r cul r bloc ro n or s pp w c s n c c .

1 = odulo

r j = n or bloc nu b r

= o l no. o s s n c c

= c c s nu b r

A gorith :

1. En r o l nu b r o bloc s n n or =
2. En r o l nu b r o c c or bloc s = b
4. En r nu b r o bloc s n on s s = c
5. C lcul nu b r o s s = b/c
6. En r n or bloc nu b r
7. F n s nu b r s c = /
8. C lcul gb nu b r s = /
9. op

Conclusion:

Post Lab Assignments :

1. Show how to associate appropriate following structure.
   Class size = 32 bits
   Union size = 2 bits
   Show the structure with number of bits.

2. Associate the consists of 10 blocks. Union consists 10 blocks of 8 bits each. How the union address of each will be ?

Expem ent No 9

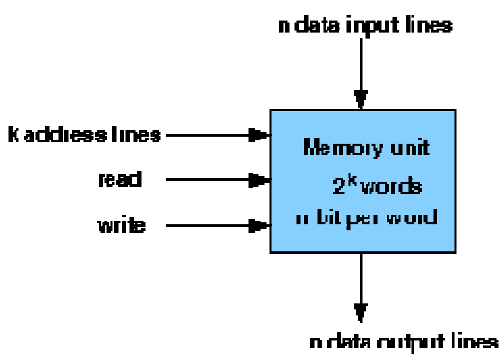
Desıgn of Me ory

Title: r te pro g o e gn e or un s.

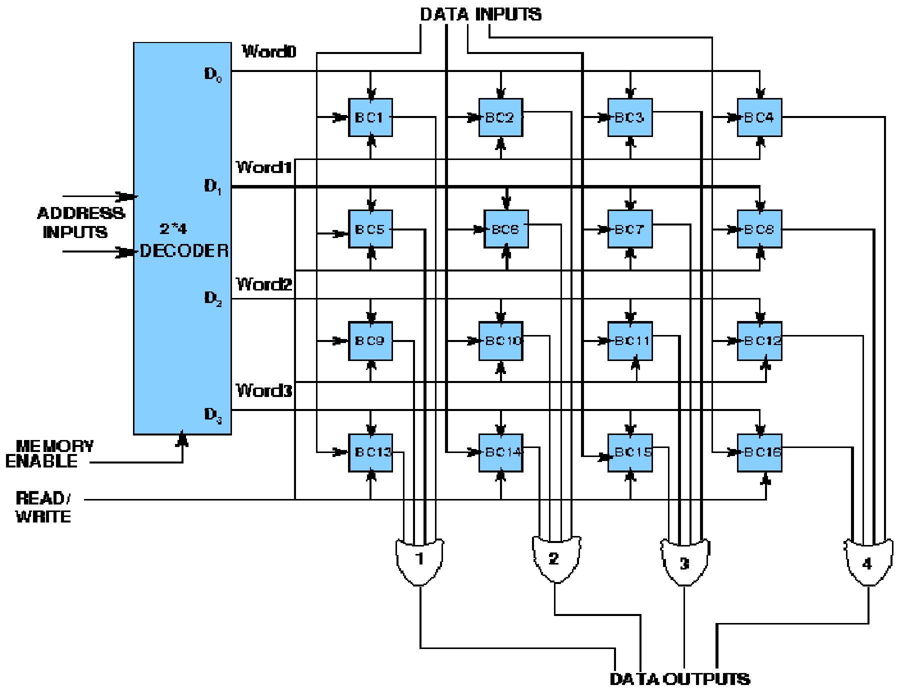
Objective: o e gn e or un s n un ers n how op r e ur n g re n w r e oper on.

References. 1. C r , cher, Z on o r n c n S w , Z , "Co pu er r g n on", F h E on, c r w - .
2. S n g, "Co pu er r g n on n Arch ec ure: De g n g or er or nce", E gh E on, e s on.

Theory. A e or un s collec on o s or g cells o g e w h ss oc e c rcu s n e e o r ns or n or on n ou o he e ce w e or cells wh ch c n be cc sse or n or on r ns er o or ro n e re r n o loc on s c lle r n o cc ss e or (A). he bloc g o e or un -



A e or w h 4 w or s n e s w o r ss l n s . he w o r ss npu s g hrou gh 2*4 eco er o s elec one o he ou w or s . he eco er s en ble w h he e or en ble npu hen he e or en bles 0, ll ou pus o he eco er re 0 n none o he e or w or s re s elec e . h he e or en ble 1, one o he ou w or s s s elec e , c e b he lue n he w o r ss l n s . nce w or h s been s elec e , he re w r e npu e er nes he oper on. he lo gc g s -



Conclusion:

Post Lab Assignments:

- What is the maximum time required before a [illegible] must be refreshed?
- What is the number of address bits for addressing 256 KB?

Experiment No 7

Replacement Algorithm

Title: Write a program to implement different replacement algorithm.

i). To implement LRU (Least Recently Used) cache algorithm.

ii). To implement FIFO (First in First out) cache algorithm.

iii). To implement LFU (Least Frequently Used) cache algorithm.

Objective: Understanding the cache replacement algorithms, how old blocks are replaced by new one when the cache is full.

References.

1. Carl Hamacher, Zvonko Vranesic and Safwat Zaky, “Computer Organization”, Fifth Edition, McGraw-Hill.
2. William Stallings, “Computer Organization and Architecture: Designing for Performance”, Eighth Edition, Pearson.

Theory: In case of a miss, a new block needs to be brought into the cache. If all the positions it may occupy are full, then the cache controller must decide which of the old blocks to overwrite. This is done by replacement algorithms so as to maximize hit ratios.

[illegible]