Computer Architecture Assignment

Fast Adders
(Carry look Ahead Adder)

Nithya Sree K
2021115071
Karthika P
2021115049

Carry look Ahead Adder:

Different types of Digital systems are constructed from very few types of basic network configurations such as AND gate, NAND gate, Or gate, etc…These elementary circuits are used over and over again in various topological combinations. In addition to performing logic, digital systems must also store binary numbers. For these memory cells, also known as FLIP-FLOP's are designed. To perform some functions such as binary addition. Hence, to perform such functions, combinations of logic gates and FLIP-FLOPs are designed over a single-chip IC. These IC's form the practical building blocks of the Digital systems. One of such building blocks used for binary addition is the Carry Look-ahead Adder.

What is a Carry Look-ahead Adder?

A digital computer must contain circuits which can perform arithmetic operations such as addition, subtraction, multiplication, and division. Among these, addition and subtraction are the basic operations whereas multiplication and division are the repeated addition and subtraction respectively.

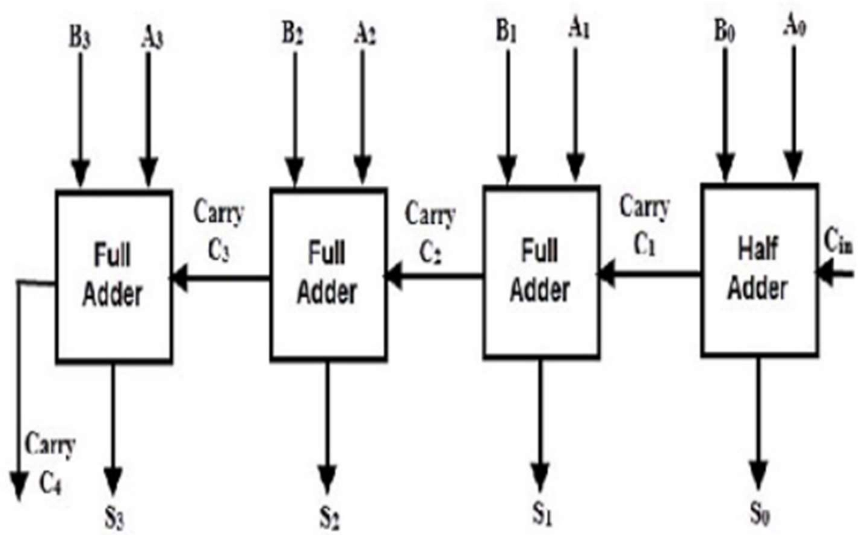
To perform these operations 'Adder circuits' are implemented using basic logic gates. Adder circuits are evolved as Half-adder, Full-adder, Ripple-carry Adder, and Carry Look-ahead Adder.

Among these Carry Look-ahead Adder is the faster adder circuit. It reduces the propagation delay, which occurs during addition, by using more complex hardware circuitry. It is designed by transforming the ripple-carry Adder circuit such that the carry logic of the adder is changed into two-level logic.

4-Bit Carry Look-ahead Adder

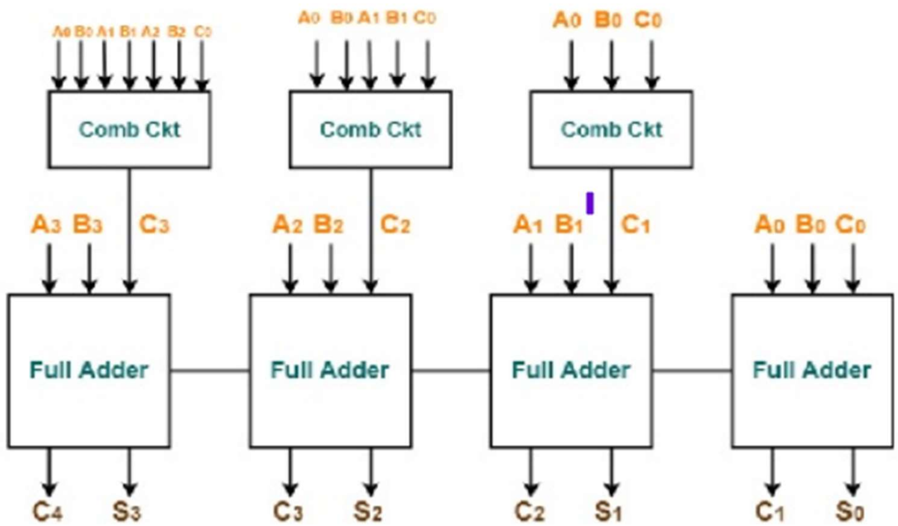
In parallel adders, carry output of each full adder is given as a carry input to the next higher-order state. Hence, these adders it is not possible to produce carry and sum outputs of any state unless a carry input is available for that state.

So, for computation to occur, the circuit has to wait until the carry bit propagated to all states. This induces carry propagation delay in the circuit.



4-bit-Ripple-Carry-Adder

The propagation delay of the adder is calculated as “the propagation delay of each gate times the number of stages in the circuit”. For the computation of a large number of bits, more stages have to be added, which makes the delay much worse. Hence, to solve this situation, Carry Look-ahead Adder was introduced.



4-bit-Carry-Look-ahead-Adder-Logic-Diagram

In this adder, the carry input at any stage of the adder is independent of the carry bits generated at the independent stages. Here the output of any stage is dependent only on the bits which are added in the previous stages and the carry input provided at the beginning stage. Hence, the circuit at any stage does not have to wait for the generation of carry-bit from the previous stage and carry bit can be evaluated at any instant of time.

Truth Table of Carry Look-ahead Adder

For deriving the truth table of this adder, two new terms are introduced – Carry generate and carry propagate. Carry generate Gi =1 whenever there is a carry Ci+1 generated. It depends on Ai and Bi inputs. Gi is 1 when both Ai and Bi are 1. Hence, Gi is calculated as Gi = Ai. Bi.

Carry propagated Pi is associated with the propagation of carry from Ci to Ci+1. It is calculated as Pi = Ai ⊕ Bi. The truth table of this adder can be derived from modifying the truth table of a full adder.

Using the Gi and Pi terms the Sum Si and Carry Ci+1 are given as below –

- $Si = Pi \oplus Gi$.
- Ci+1 = Ci.Pi +Gi.

Therefore, the carry bits C1, C2, C3, and C4 can be calculated as

- C1 = C0.P0+G0.
- C2 = C1.P1+G1 = (C0.P0+G0).P1+G1.
- C3 = C2.P2+G2 = (C1.P1+G1).P2+G2.
- C4 = C3.P3+G3 = C0.P0.P1.P2.P3 + P3.P2.P1.G0 + P3.P2.G1 + G2.P3 + G3.

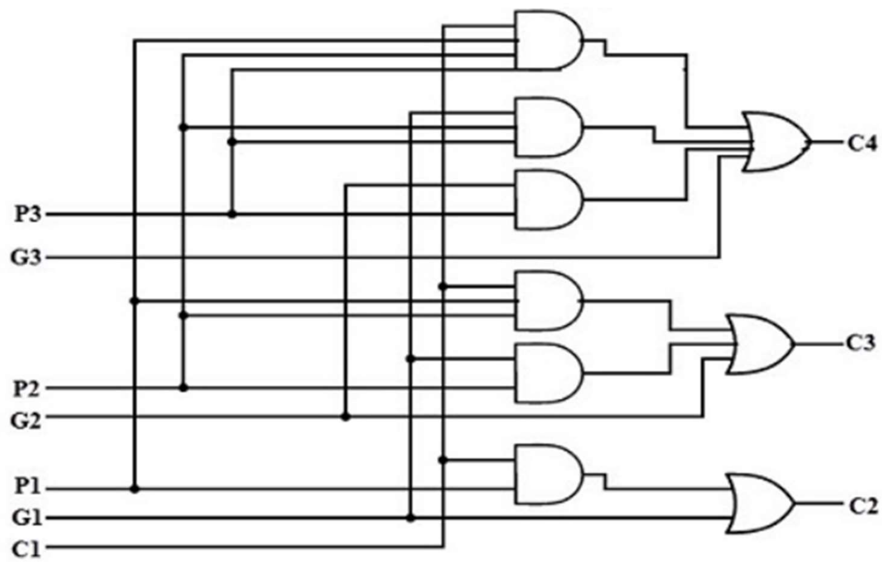
It can be observed from the equations that carry Ci+1 only depends on the carry C0, not on the intermediate carry bits.

A	B	Ci	C i +1	Condition
0	0	0	0	No carry generate
0	0	1	0	
0	1	0	0	
0	1	1	1	No carry propagate
1	0	0	0	
1	0	1	1	
1	1	0	1	Carry generate
1	1	1	1	

Carry-Look-ahead-Adder-Truth-Table

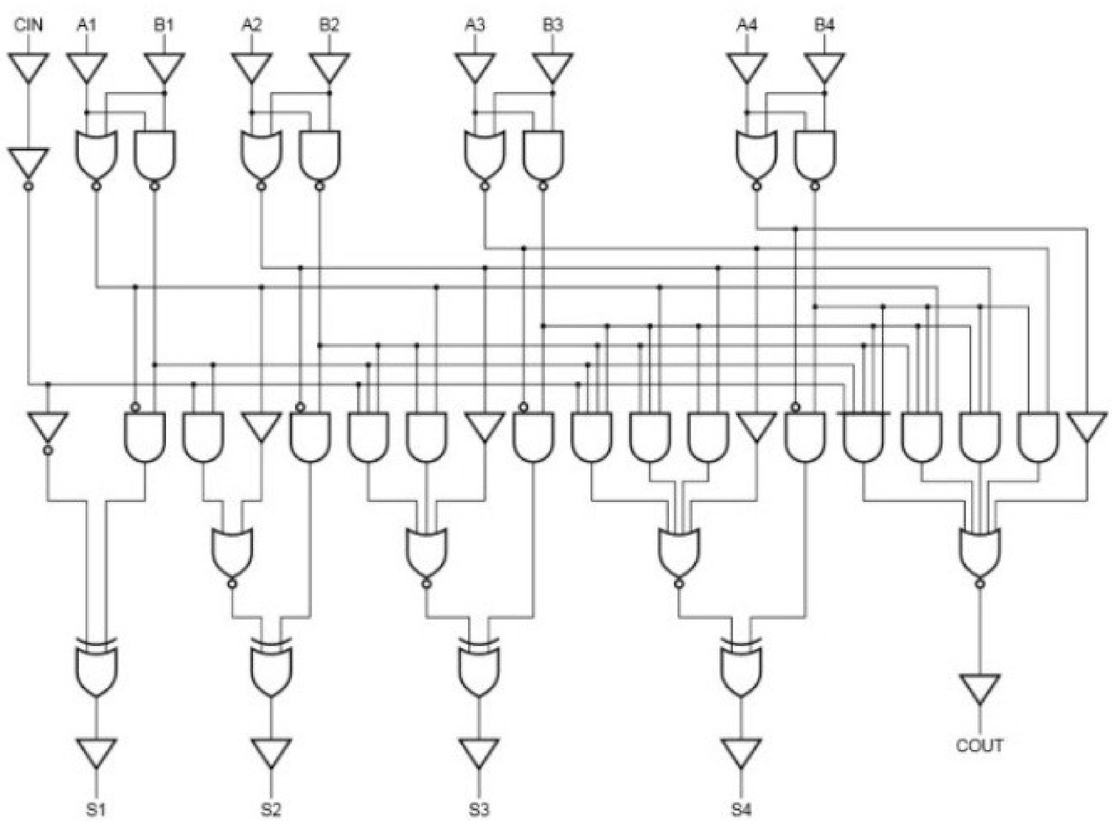
Circuit Diagram

The above equations are implemented using two-level combinational circuits along with AND, OR gates, where gates are assumed to have multiple inputs.



Carry-Output-Generation-Circuit-of-Carry-Look-ahead-Adder

The Carry Look-ahead Adder circuit for 4-bit is given below.



8-bit and 16-bit Carry Look-ahead Adder circuits can be designed by cascading the 4-bit adder circuit with carry logic.

C program to implement Carry look Ahead Adder

```
/* C Program For Implementation Of Look Ahead Carry Adder */

#include <stdio.h>

#include <math.h>

#include <stdlib.h>

#include <string.h>

int get1(int a)

{

char ch='B';

if(a==1)

ch='A';

do

{

printf("\n\tENTER VALUE OF %c:",ch);

scanf("%d",&a);

if(a<=0)

printf("\n\t\t!INVALID NUMBER.ENTER VALUE (0< A)!");

}while(a<=0);

return(a);

}

int and(int a,int b)
```

```
{
int c;
if(a< b)
c=a;
else
c=b;
return (c);
}
int or(int a,int b)
{
int x;
if(a>b)
x=a;
else
x=b;
return x;
}

int exor(int a,int b)
{
int x;
```

```
if(a==b)
x=0;
else
x=1;
return x;
}
void add()
{
int i=7,A,B,a,b,cin,num;
int n1[8],n2[8],cg[8],cp[8],sum[8];
for(i=0;i<=7;i++)
{
n1[i]=0; // Num 1
n2[i]=0; // Num 2
cg[i]=0; // Gi
cp[i]=0; // Pi
sum[i]=0; // Sum
}
A = a = get1(1);
B = b = get1(0);
i=7;
```

```
do
{
n1[i]=a%2;
a=a/2;
n2[i]=b%2;
b=b/2;
i--;
}while((a!=0)||(b!=0));
i=0;
printf("\n\t\t Binary Form");
printf("\n\t A = %d : ",A);
for(i=0;i<=7;i++)
printf("%d ",n1[i]);
printf("\n\t B = %d : ",B);
for(i=0;i<=7;i++)
printf("%d ",n2[i]);
cin=0;
for(i=7;i>=0;i--)
{

sum[i]=exor(cin,exor(n1[i],n2[i])); // Sum Pi (+) Bi
```

```
cg[i]=and(n1[i],n2[i]); // Gi = Ai . Bi

cp[i]=or(n1[i],n2[i]); // Pi = Ai (+) Bi

cin=or(cg[i],and(cp[i],cin)); // Cin =Gi + PiCi

}

printf("\n\n\t\t SUM: ");

num=0;

for(i=0;i<=7;i++)

{

printf(" %d",sum[i]);

num=num + (sum[i]*pow(2,7-i));

}

printf("\n\n\t\t SUM: %d + %d= %d\n",A,B,num);

printf("\t\t The Carry Is : %d\n\n",cin);

}

void main()

{

int ch,a,b,c,d;

while(1)

{

M: printf("******** MENU FOR LOOK AHEAD CARRY ADDER ********");

printf("\n\t\t1.ADDITION OF TWO NUMBER");
```

```
printf("\n\t\t2.EXIT\n");

printf("**********************************************");

printf("\n\t\tEnter Your Option:");

scanf("%d",&ch);

switch(ch)

{

case 1:

add();

break;

case 2:

exit(0);

break;

default:

printf("ERROR!!!!!!!!! INVALID ENTRY...\n");

printf("Back To Main Menu\n\n");

goto M;

}

}

}
```

Output:

Output　　Clear

```
/tmp/NpfF3xcCU8.o
******** MENU FOR LOOK AHEAD CARRY ADDER ********
        1.ADDITION OF TWO NUMBER
        2.EXIT
*************************************************
        Enter Your Option:1
        ENTER VALUE OF A:12
        ENTER VALUE OF B:5
        Binary Form
     A = 12 : 0 0 0 0 1 1 0 0
     B = 5 : 0 0 0 0 0 1 0 1

         SUM:  0 0 0 1 0 0 0 1

         SUM: 12 + 5= 17
         The Carry Is : 0

******** MENU FOR LOOK AHEAD CARRY ADDER ********
        1.ADDITION OF TWO NUMBER
        2.EXIT
*************************************************
        Enter Your Option:1
        ENTER VALUE OF A:3
        ENTER VALUE OF B:-2
        !INVALID NUMBER.ENTER VALUE (0< A)!
    ENTER VALUE OF B:5
```

Output　　Clear

```
******** MENU FOR LOOK AHEAD CARRY ADDER ********
        1.ADDITION OF TWO NUMBER
        2.EXIT
*************************************************
        Enter Your Option:1
        ENTER VALUE OF A:3
        ENTER VALUE OF B:-2
        !INVALID NUMBER.ENTER VALUE (0< A)!
    ENTER VALUE OF B:5
    Binary Form
     A = 3 : 0 0 0 0 0 0 1 1
     B = 5 : 0 0 0 0 0 1 0 1

SUM:  0 0 0 0 1 0 0 0

         SUM: 3 + 5= 8
         The Carry Is : 0

******** MENU FOR LOOK AHEAD CARRY ADDER ********
        1.ADDITION OF TWO NUMBER
        2.EXIT
*************************************************
        Enter Your Option:1
        ENTER VALUE OF A:11
        ENTER VALUE OF B:52
        Binary Form
```

Output Clear

```
SUM:  0 0 0 0 1 0 0 0

        SUM: 3 + 5= 8
        The Carry Is : 0

******** MENU FOR LOOK AHEAD CARRY ADDER ********
       1.ADDITION OF TWO NUMBER
       2.EXIT
*************************************************
       Enter Your Option:1
       ENTER VALUE OF A:11
       ENTER VALUE OF B:52
       Binary Form
     A = 11 : 0 0 0 0 1 0 1 1
     B = 52 : 0 0 1 1 0 1 0 0

        SUM:  0 0 1 1 1 1 1 1

        SUM: 11 + 52= 63
        The Carry Is : 0

******** MENU FOR LOOK AHEAD CARRY ADDER ********
       1.ADDITION OF TWO NUMBER
       2.EXIT
*************************************************
       Enter Your Option:2
```

Mips Code:

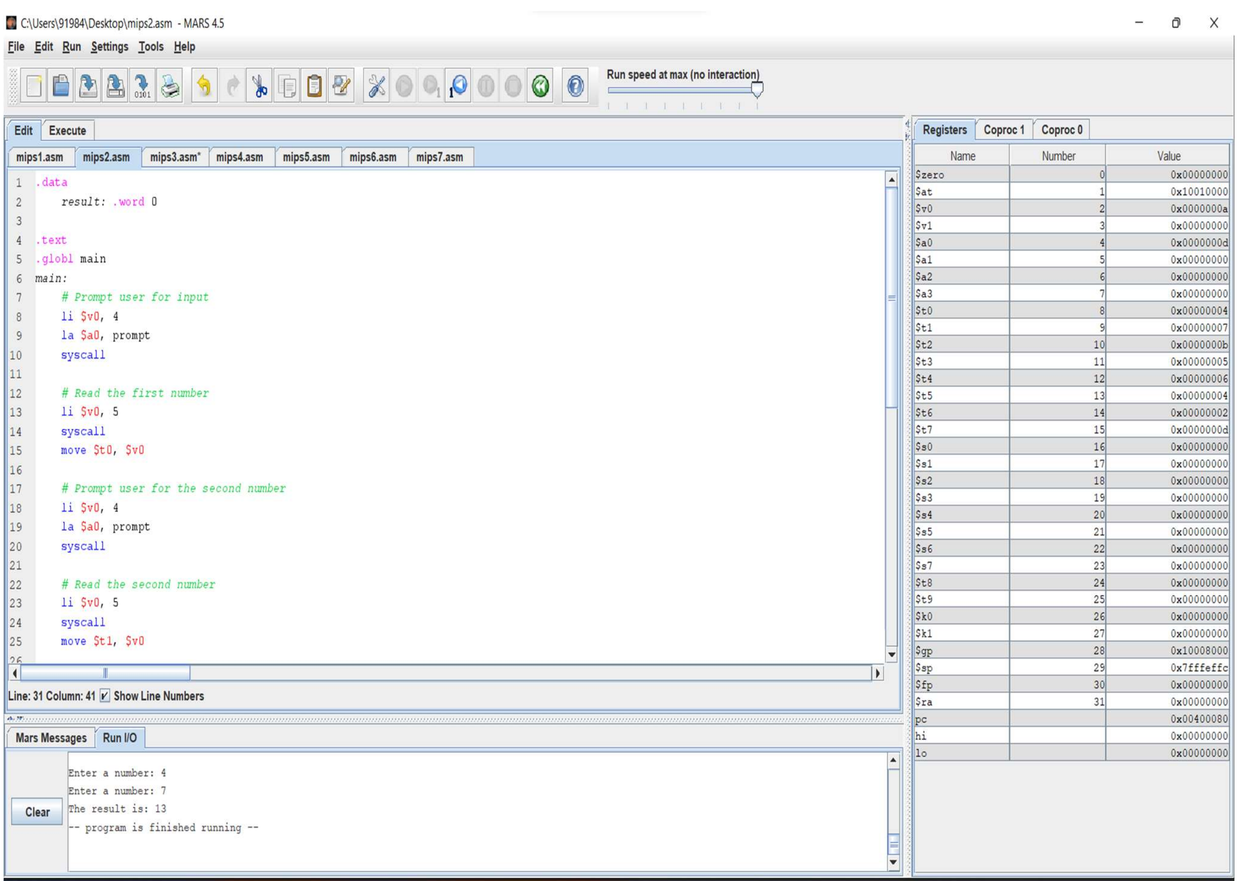
```
.data

    result: .word 0



.text

.globl main

main:

    # Prompt user for input
```

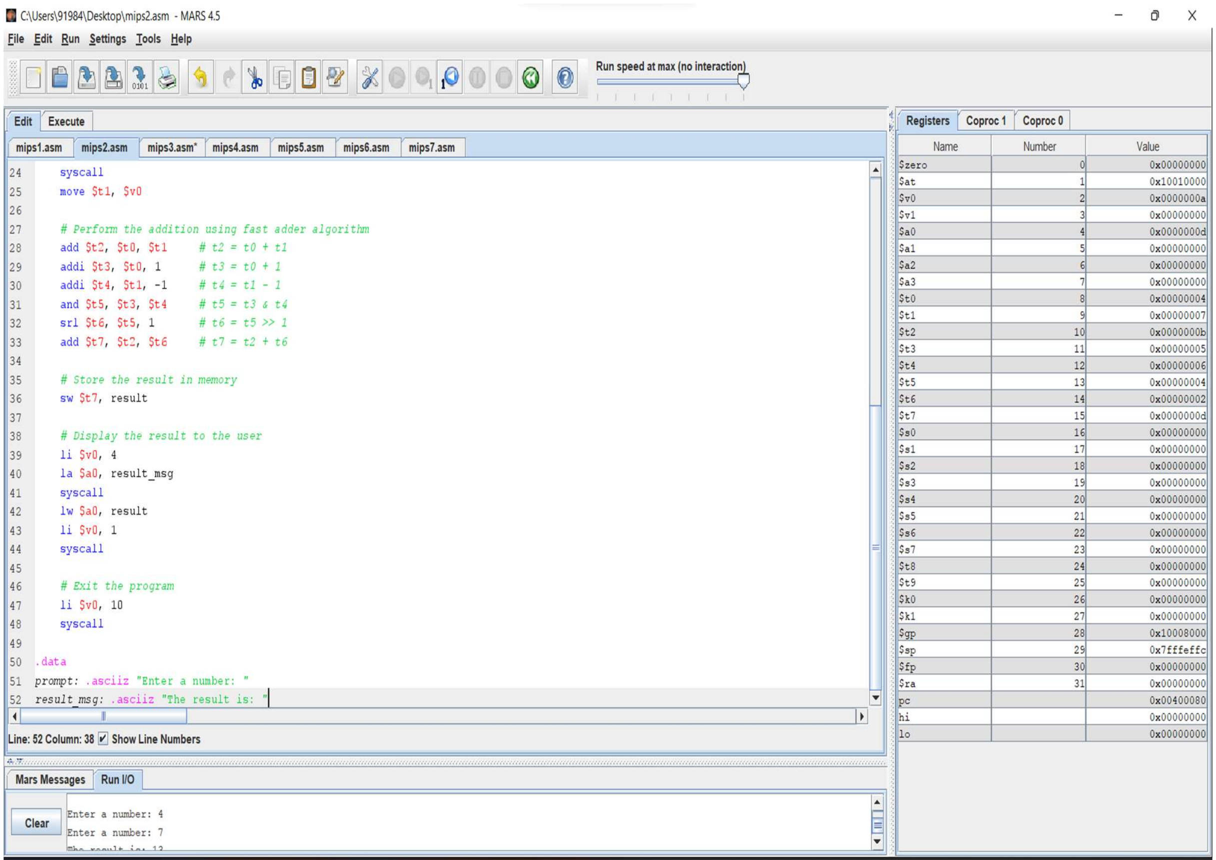
```
  li $v0, 4
la $a0, prompt
syscall

# Read the first number
li $v0, 5
syscall
move $t0, $v0

# Prompt user for the second number
li $v0, 4
la $a0, prompt
syscall

# Read the second number
li $v0, 5
syscall
move $t1, $v0
```

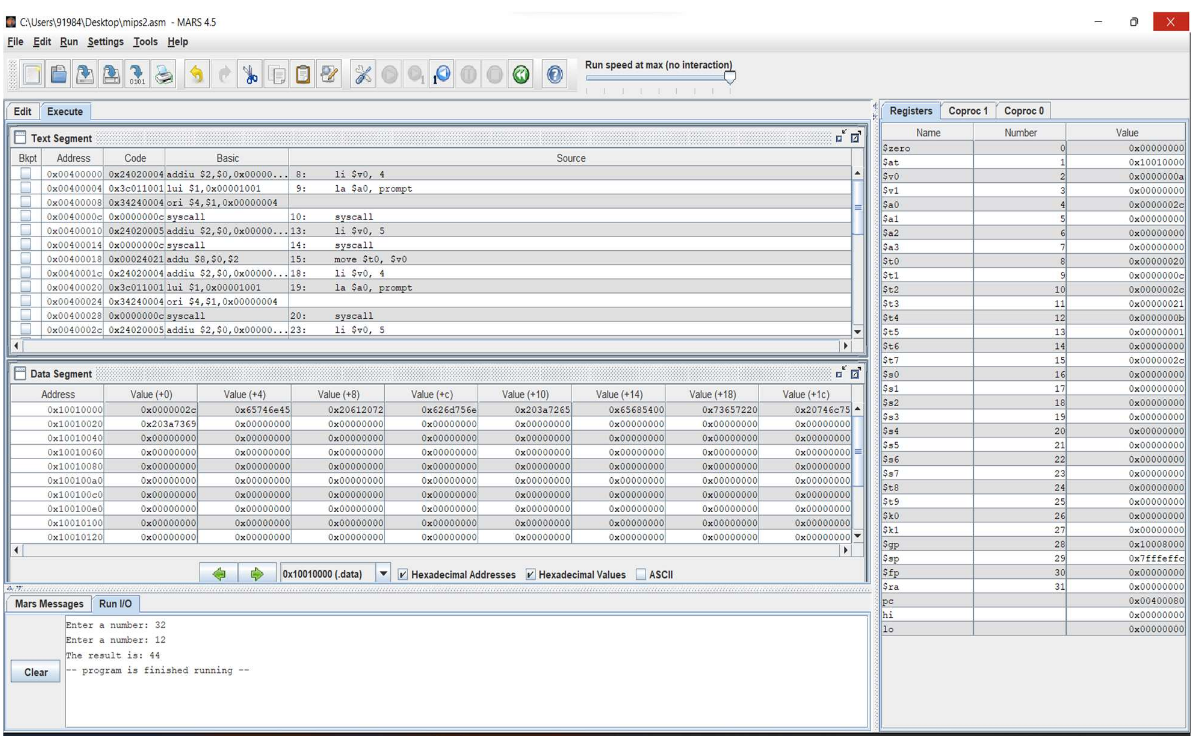
```
# Perform the addition using fast adder algorithm
add $t2, $t0, $t1     # t2 = t0 + t1
addi $t3, $t0, 1      # t3 = t0 + 1
addi $t4, $t1, -1     # t4 = t1 - 1
and $t5, $t3, $t4     # t5 = t3 & t4
srl $t6, $t5, 1       # t6 = t5 >> 1
add $t7, $t2, $t6     # t7 = t2 + t6

# Store the result in memory
sw $t7, result

# Display the result to the user
li $v0, 4
la $a0, result_msg
syscall
lw $a0, result
li $v0, 1
syscall
```

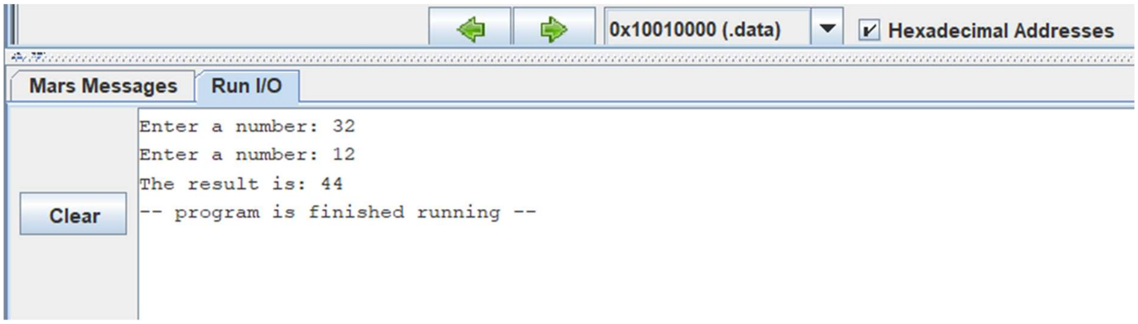
```
# Exit the program

li $v0, 10

syscall


.data

prompt: .asciiz "Enter a number: "

result_msg: .asciiz "The result is: "
```

OUTPUT:




C:\Users\91984\Desktop\mips2.asm - MARS 4.5
File Edit Run Settings Tools Help
Run speed at max (no interaction)
Edit Execute
mips1.asm mips2.asm mips3.asm* mips4.asm mips5.asm mips6.asm mips7.asm
syscall
move $t1, $v0
# Perform the addition using fast adder algorithm
add $t2, $t0, $t1 # t2 = t0 + t1
addi $t3, $t0, 1 # t3 = t0 + 1
addi $t4, $t1, -1 # t4 = t1 - 1
and $t5, $t3, $t4 # t5 = t3 & t4
srl $t6, $t5, 1 # t6 = t5 >> 1
add $t7, $t2, $t6 # t7 = t2 + t6
# Store the result in memory
sw $t7, result
# Display the result to the user
li $v0, 4
la $a0, result_msg
syscall
lw $a0, result
li $v0, 1
syscall
Exit the program
li $v0, 10
syscall
.data
prompt: .asciiz "Enter a number: "
result_msg: .asciiz "The result is: "
Line: 52 Column: 38 Show Line Numbers
Mars Messages Run I/O
Clear
Enter a number: 4
Enter a number: 7
Registers Coproc 1 Coproc 0
Name Number Value


C:\Users\91984\Desktop\mips2.asm - MARS 4.5
File Edit Run Settings Tools Help
Run speed at max (no interaction)
Edit Execute
Text Segment
Bkpt Address Code Basic Source
Data Segment
Address Value (+0) Value (+4) Value (+8) Value (+c) Value (+10) Value (+14) Value (+18) Value (+1c)
0x10010000 (.data) Hexadecimal Addresses Hexadecimal Values ASCII
Mars Messages Run I/O
Clear
Enter a number: 32
Enter a number: 12
The result is: 44
-- program is finished running --
Registers Coproc 1 Coproc 0
Name Number Value



Advantages of Carry Look Ahead Adder

The advantages of carry look ahead adder are-

- It generates the carry-in for each full adder simultaneously.
- It reduces the propagation delay.

Disadvantages of Carry Look Ahead Adder-

The disadvantages of carry look ahead adder are-

- It involves complex hardware.
- It is costlier since it involves complex hardware.
- It gets more complicated as the number of bits increases.

Applications

The **carry lookahead adder applications** are:

- Carry lookahead adders operating with high speed are employed as integrated circuits so that it is simple to integrate adder in many circuits. Also, the increase in the count of gates is even moderate when implemented for higher bits.

- When CLA’s are used for high-bit calculations, the device offers more speed whereas the circuit complexity also increases. Usually, these are used for 4-bit modules so that they are integrated together for high-bit computations.
- On a regular basis, carry-lookahead adders are used in boolean computations.