Inline Assembly Program

```
asm ( assembler template
    : output operands                /* optional */
    : input operands                 /* optional */
    : list of clobbered registers /* optional */
    );
```

An Example

```
int main() {
 int n, i, fact = 1 ;

 printf("Enter a natural number:") ;
 scanf("%d",&n);

 for(i=1; i<=n; ++i) fact = fact*i ;

 printf("\n%d! = %d\n",n,fact);
}
```

With inline Assembly

```
int main() {
 int n, fact ;

 printf("Enter a natural number:") ;
 scanf("%d",&n);
```

for-loop Replaced

```
asm("movl $1, %%eax # fact\n\t"
    "movl %%eax, %%edx # i\n\t"
    ".MyL2:            # Label\n\t"
    "cmpl %1, %%edx # if i <= n\n\t"
    "jg .MyL1          # if i>n goto .MyL1\n\t"
    "imull %%edx, %%eax # fact = fact*i\n\t"
    "incl %%edx        # ++i\n\t"
    "jmp .MyL2         # goto .MyL2 \n\t"
    ".MyL1:            # Label\n\t"
    "movl %%eax, %0 # output fact\n\t"
    :"=r"(fact)
    :"r"(n)
```

```
 :"%eax", "%edx"
);
```

End of Code

```
 printf("\n%d! = %d\n",n,fact);
}
```

Explanation

- **:"r"(n) - the input to the code is coming from the C variable n and will be copied in a register not mentioned explicitely. The register will be refered as %1.**
- **:"=r"(fact) - the output from the code will be copied to the C variable fact from the register %0 (not named explicitely).**

Explanation

- **:"%eax", "%edx" - the two registers eax and edx are used by the assembly code. This is to inform the compiler to save these registers before entering the code.**

Explanation

- **"movl $1, %%eax # fact\n\t" - The \n\t is to put the next assembly code in the next line after a tab. The # (sharp) is for comment.**
- **movl $1, %%eax is eax = 1.**
- **movl %%eax, %%edx is edx = eax.**
- **.MyL2: is a label.**
- **cmpl %1, %%edx is compare n and edx (i).**

Explanation

- **jg .MyL1 is if i > n goto .MyL1.**
- **imull %%edx, %%eax is eax = eax*edx.**
- **incl %%edx is ++edx.**
- **jmp .MyL2 is goto .MyL2.**
- **.MyL1: is label .MyL1.**
- **movl %%eax, %0 is copy to output.**

Inside .s File

```
  movl  -4(%ebp), %ecx
#APP
  movl $1, %eax # fact
  movl %eax, %edx # i
  .MyL2:          # Label
  cmpl %ecx, %edx # if i <= n
  jg .MyL1        # if i>n goto .MyL1
  imull %edx, %eax # fact = fact*i
  incl %edx      # ++i
  jmp .MyL2       # goto .MyL2
  .MyL1:          # Label
  movl %eax, %ecx # output fact
```

```
#NO_APP
```

2nd Example: Array Access

```
int main() {
 int a[20], n, i, sum ;

 printf("Enter a +ve integer\n") ;
 scanf("%d", &n) ;
 printf("Enter %d +ve integers\n", n) ;
 for(i=0; i<n; ++i) scanf("%d", &a[i]) ;
 printf("The %d +ve integers are:\n", n) ;
 for(i=0; i<n; ++i) printf("%d ", a[i]) ;
 printf("\nThe sum of %d +ve integers is :", n) ;

// for(sum = i = 0; i<n; ++i) sum = sum + a[i] ;
```

2nd Example: Array Access

```
asm(
     "movl $0, %%eax # i = 0 \n\t"
     "movl %%eax, %%edx # sum = i \n\t"
     ".MyL1:                      \n\t"
     "cmpl %2, %%eax # if i < n\n\t"
     "jge .MyL2      # if i >= n goto .MyL2\n\t"
     "addl 0(%1,%%eax,4), %%edx # sum=sum+a[i] \n\t"
     "incl %%eax \n\t"
     "jmp .MyL1 \n\t"
     ".MyL2:    \n\t"
     "movl %%edx, %0 \n\t"
    :"=r"(sum)
```

```
     :"r"(a),"r"(n)
     :"%eax", "%edx"
    );

 printf(" %d\n", sum) ;
}
```

2nd Example: Array Access

```
$ a.out
Enter a +ve integer
5
Enter 5 +ve integers
1 5 2 4 3
The 5 +ve integers are:
1 5 2 4 3
The sum of 5 +ve integers is : 15
$
```

3rd Example: Unsigned Multiplication

```
/**** Mutiplication of two unsigned numbers *****/
int main()
{
 int a, b, c ;

 printf("Enter two +ve integers\n") ;
 scanf("%d%d", &a, &b) ;
```

3rd Example: Unsigned Multiplication

```
asm(
     "movl $0, %%eax   # Hi = 0 \n\t"
     "movl $32, %%edx  # count = 32 \n\t"
     ".MyL1:           # Label \n\t"
     "btl $0, %1       # Test b0 of %1 (Lo) \n\t"
     "jae .MyL2        # if CF = 0, goto .MyL2 \n\t"
     "addl %2, %%eax   # Hi = Hi + %2  \n\t"
     ".MyL2:           # Label \n\t"
     "rcrl $1, %%eax   # Rotate Hi through CF \n\t"
     "rcrl $1, %1      # Rotate Lo through CF \n\t"
     "decl %%edx       # --count \n\t"
     "jnz .MyL1        # goto .MyL1 \n\t"
```

3rd Example: Unsigned Multiplication

```
      :"=r"(c)
      :"0"(a),"r"(b)
      :"%eax", "%edx"
    ) ;
 printf("%d * %d = %d\n", a, b, c) ;
}
```

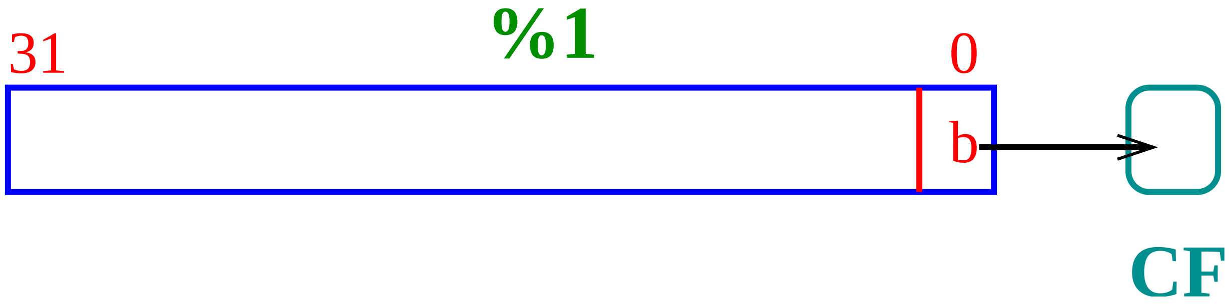

Figure 1: **btl $0, %1**

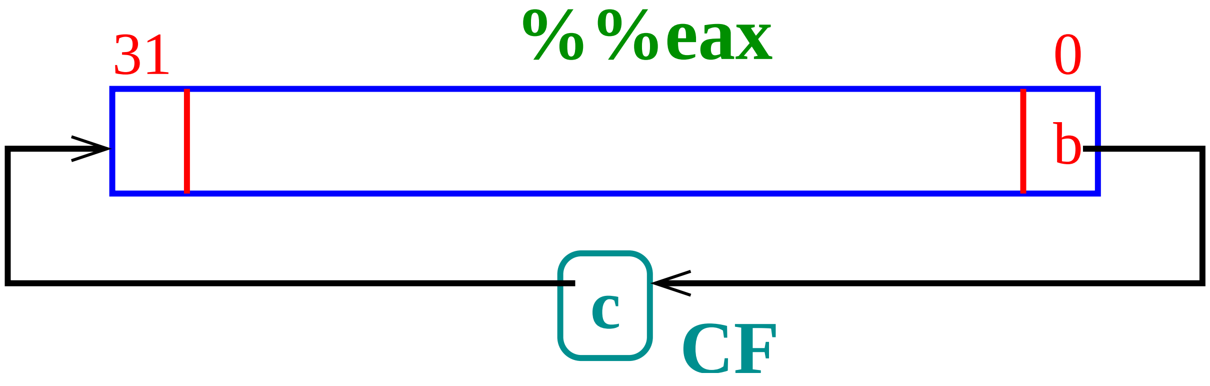

Figure 2: **rcrl $1, %%eax**

Manual

- **GCC-Inline-Assembly-HOWTO: http://www.ibiblio.org/gferg/ldp/GCC-Inline-Assembly-HOWTO.html#ss5.3**

Assignment-II

Write a C program with inline assembly language component to find the mean and the median of a set of n integers.

- **Read n integers in a 1-D array.**
- **Compute mean and median using inline assembly language program of Intel x86.**
- **Print the set elements, the mean and the median.**

Use printf and scanf for IO.