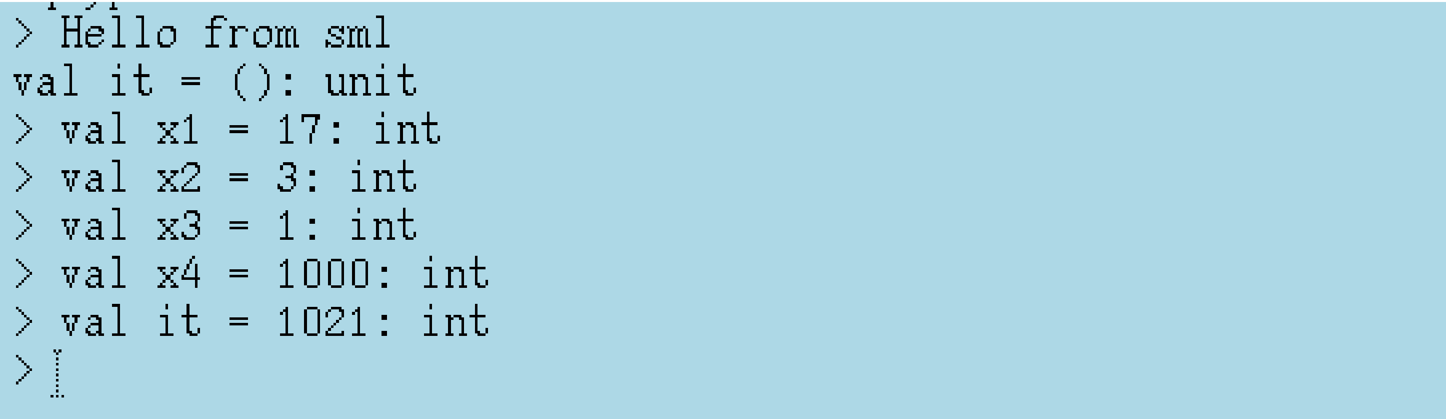
cs118

SML

```
open TextIO;
output(stdOut,"Hello from sml\n");
val x1 = 17;
val x2 = 3;
val x3 =1;
val x4 = 1000;
x1 + x2 + x3 + x4;
```

In a functional language we evaluate expression rather than change the store. Our expected result from adding those 4 values is 1021:



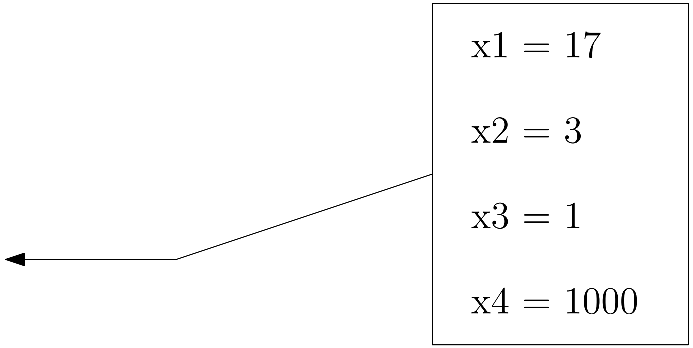
```
> Hello from sml
val it = (): unit
> val x1 = 17: int
> val x2 = 3: int
> val x3 = 1: int
> val x4 = 1000: int
> val it = 1021: int
> 
```

The following code is a simple C program to display hello

C source code written to file lab.c

```
#include <stdio.h>
int main()
{
  printf("Hello\n");
  int x1 =17;
  int x2 = 3;
  int x3 = 1;
  int x4 = 1000;
  x1 += x2;
  x1 += x3;
  x1 += x4;
  printf("The value of x1 is %d\n",x1);
  return 0;
}
```

```
#include <stdio.h>
int main()
{
  printf("Hello\n");
  int x1 =17;
  int x2 = 3;
  int x3 = 1;
  int x4 = 1000;
  x1 += x2;
  x1 += x3;
  x1 += x4;
  printf("The value of x1 is %d\n",x1);
  return 0;
}
```



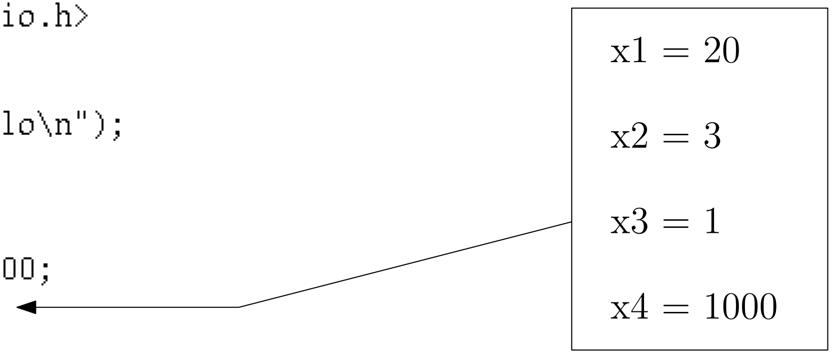
$x1 = 17$

$x2 = 3$

$x3 = 1$

$x4 = 1000$

```
#include <stdio.h>
int main()
{
  printf("Hello\n");
  int x1 =17;
  int x2 = 3;
  int x3 = 1;
  int x4 = 1000;
  x1 += x2;
  x1 += x3;
  x1 += x4;
  printf("The value of x1 is %d\n",x1);
  return 0;
}
```



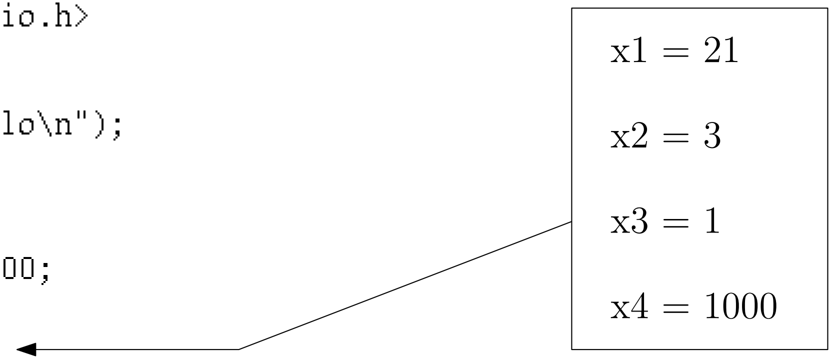
$x1 = 20$

$x2 = 3$

$x3 = 1$

$x4 = 1000$

```
#include <stdio.h>
int main()
{
  printf("Hello\n");
  int x1 =17;
  int x2 = 3;
  int x3 = 1;
  int x4 = 1000;
  x1 += x2;
  x1 += x3;
  x1 += x4;
  printf("The value of x1 is %d\n",x1);
  return 0;
}
```



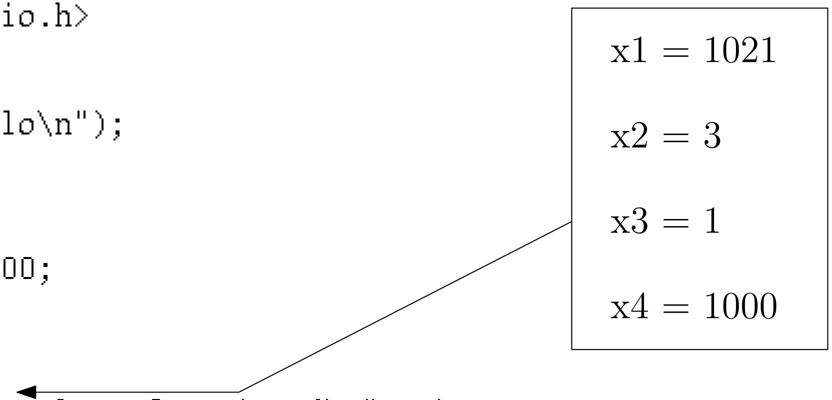
$x1 = 21$

$x2 = 3$

$x3 = 1$

$x4 = 1000$

```
#include <stdio.h>
int main()
{
  printf("Hello\n");
  int x1 =17;
  int x2 = 3;
  int x3 = 1;
  int x4 = 1000;
  x1 += x2;
  x1 += x3;
  x1 += x4;
  printf("The value of x1 is %d\n",x1);
  return 0;
}
```



x1 = 1021

x2 = 3

x3 = 1

x4 = 1000

The following code is a simple ASM program to display Hello This command is used to create the executable linking to the dynamic system libraries:
`as -gstabs -o lab.o lab.s ld -dynamic-linker /lib/ld-linux.so.2 -o labasm lab.o -lc`la

asm source code written to file lab.s

```
.data #Where to list any memory storage you will need for data
fmt: .string "Hello from asm\n"
fmt2: .string "x1 = %d\n"
.text #where the program instructions live
.globl _start #where program starts, same as main function in C
_start: #define the value of _start label
```

Display message about program

asm source code appended to file lab.s

```
push $fmt
call printf
add $4,%esp
```

Initialize registers to some values

asm source code appended to file lab.s

```
mov $17,%eax     #eax = 17
mov $3,%ebx      #ebx = 3
mov $1,%ecx      #ecx = 1
mov $1000,%edx   #edx = 1000
```

Accumulate the values into the eax register asm source code appended to file lab.s

```
add %ebx,%eax
add %ecx,%eax
add %edx,%eax
```

Display the result

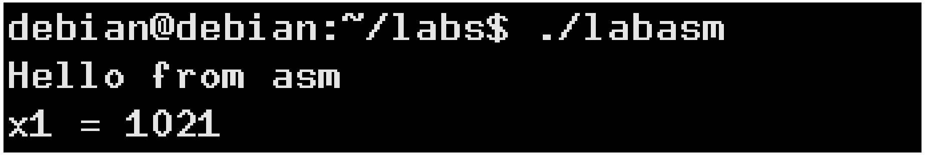
asm source code appended to file lab.s

```
push %eax
push $fmt2
call printf
add $8,%esp
```

Exit program by calling Linux OS command exit

asm source code appended to file lab.s

```
mov $1,%eax #1 is the number of exit system call, require status code in %ebx
mov $0,%ebx #0 is returned to the system
int $0x80
```



```
debian@debian:~/labs$ ./labasm
Hello from asm
x1 = 1021
```