Credits and Disclaimers

The examples and discussion in the following slides have been adapted from a variety of sources, including:

Chapter 3 of Computer Systems 2nd Edition by Bryant and O'Hallaron
x86 Assembly/GAS Syntax on WikiBooks
(http://en.wikibooks.org/wiki/X86_Assembly/GAS_Syntax)
Using Assembly Language in Linux by Phillip ??
(http://asm.sourceforge.net/articles/linasm.html)

The C code was compiled to assembly with `gcc` version 4.5.2 on Ubuntu Linux.

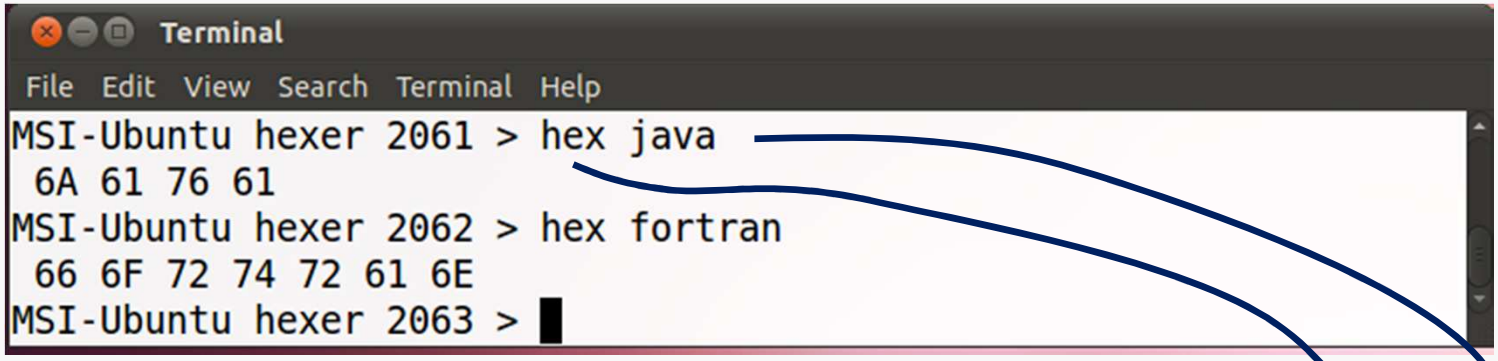
Unless noted otherwise, the assembly code was generated using the following command line:

```
gcc -S -m32 -O0 file.c
```

AT&T assembly syntax is used, rather than Intel syntax, since that is what the `gcc` tools use.

Shell Side to C

C programs can receive command-line arguments from the shell:



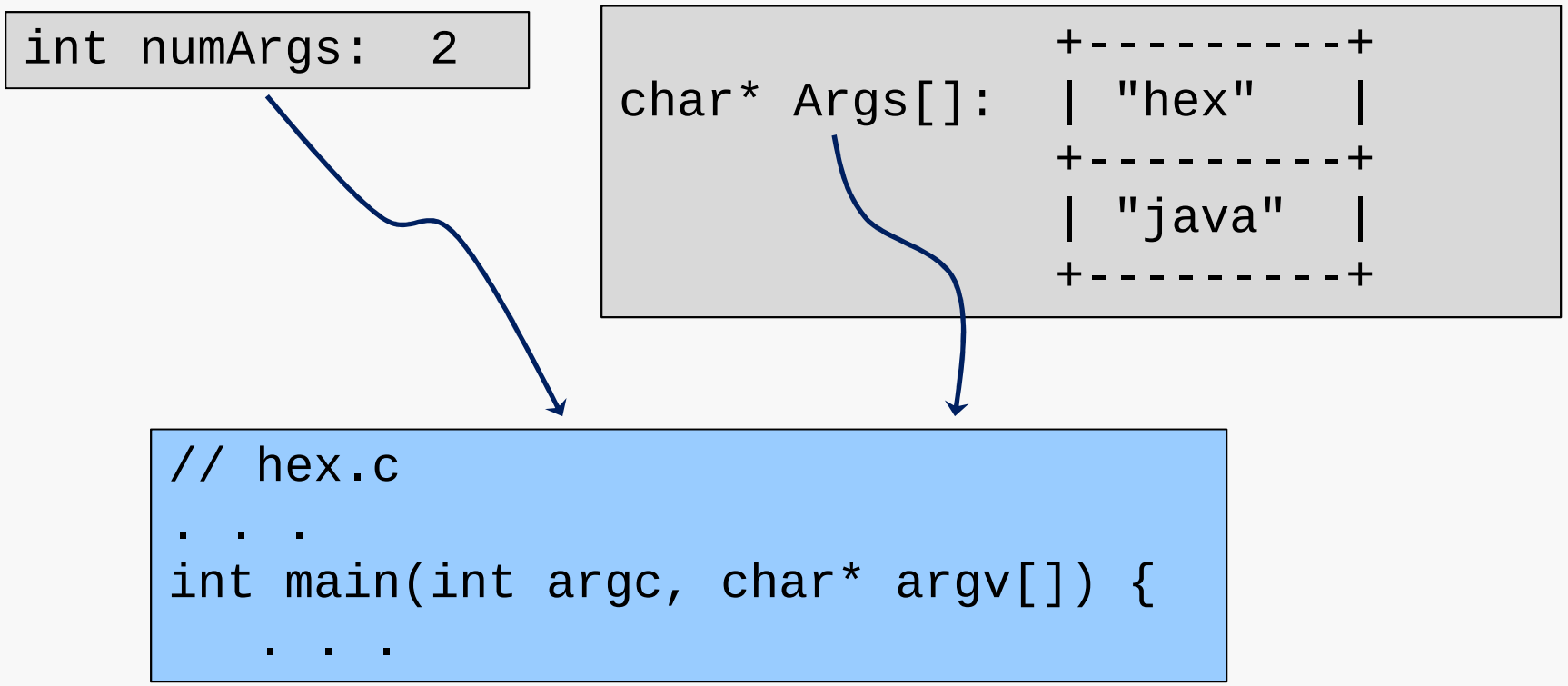
The shell initializes an integer variable and an array of C-style strings:

```
int numArgs:  2
```

```
                    +---------+
char* Args[]:       | "hex"   |
                    +---------+
                    | "java"  |
                    +---------+
```

main() Interface

These arguments are passed as parameters to `main()`:



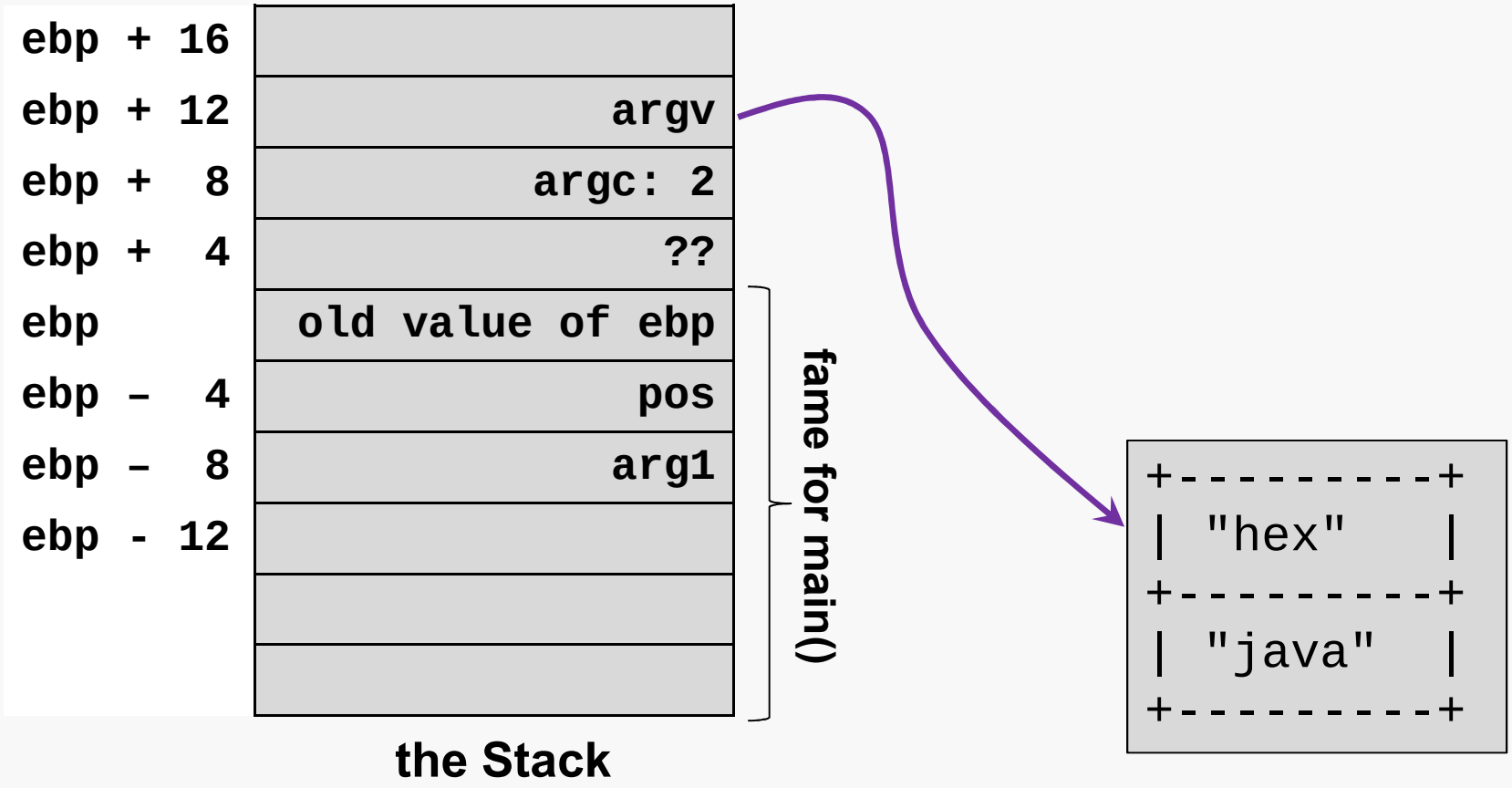
So, the C program can now check the number of command-line "tokens" and process them as needed.

C Code

```
// hex.c
#include <stdio.h>

int main(int argc, char* argv[]) {

   if ( argc != 2 ) {     // check # of params
      return 1;
   }

   char* arg1 = argv[1];  // slap handle on 2nd one
   int pos = 0;

   // print ASCII codes of characters, in hex format:
   while ( arg1[pos] != '\0' ) {
      printf(" %X", (unsigned char) arg1[pos]);
      pos++;
   }
   printf("\n");
   return 0;
}
```

The Stack

These arguments are passed as to `main()` via the stack:



Assembly View

```
// hex.c
int main(int argc, char* argv[]) {
    . . .
 char* arg1 = argv[1];
    . . .
}
```

```
. . .
cmpl   $2,          8(%ebp)
. . .
movl   12(%ebp), %eax
addl   $4,          %eax
movl   (%eax),      %eax
movl   %eax,        -8(%ebp)
. . .
```