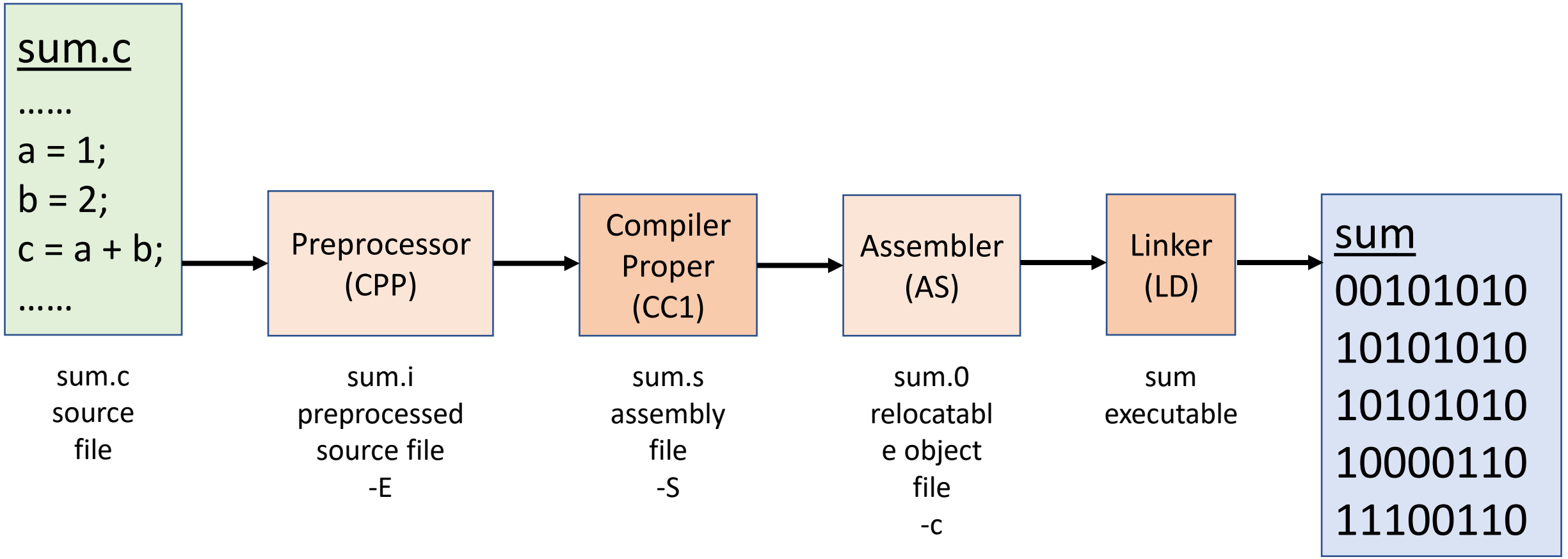
CS 354
Machine Organization and Programming
Lecture 03

Michael Doescher
Summer 2020

Review
Java to C
Memory Model
Arrays and Pointers

Build Process



Corrections

-pass-exit-codes

Normally the **gcc** program will exit with the code of 1 if any phase of the compiler returns a non-success return code. If you specify **-pass-exit-codes**, the **gcc** program will instead return with numerically highest error produced by any phase that returned an error indication. The C, C ++ , and Fortran frontends return 4, if an internal compiler error is encountered.

If you only want some of the stages of compilation, you can use **-x** (or filename suffixes) to tell **gcc** where to start, and one of the options **-c**, **-S**, or **-E** to say where **gcc** is to stop. Note that some combinations (for example, **-x cpp-output -E**) instruct **gcc** to do nothing at all.

-c

Compile or assemble the source files, but do not link. The linking stage simply is not done. The ultimate output is in the form of an object file for each source file.

By default, the object file name for a source file is made by replacing the suffix **.c**, **.i**, **.s**, etc., with **.o**.

Unrecognized input files, not requiring compilation or assembly, are ignored.

-S

Stop after the stage of compilation proper; do not assemble. The output is in the form of an assembler code file for each non-assembler input file specified.

By default, the assembler file name for a source file is made by replacing the suffix **.c**, **.i**, etc., with **.s**.

Input files that don't require compilation are ignored.

-E

Stop after the preprocessing stage; do not run the compiler proper. The output is in the form of preprocessed source code, which is sent to the standard output.

Input files which don't require preprocessing are ignored.

Corrections

-Wall

This enables all the warnings about constructions that some users consider questionable, and that are easy to avoid (or modify to prevent the warning), even in conjunction with macros. This also enables some language-specific warnings described in **C ++ Dialect Options** and **Objective-C and Objective-C ++ Dialect Options**.

-Wall turns on the following warning flags:

-Waddress -Warray-bounds (only with **-O2**) **-Wc++0x-compat -Wchar-subscripts -Wimplicit-int -Wimplicit-function-declaration -Wcomment -Wformat -Wmain** (only for C/ObjC and unless **-ffreestanding**) **-Wmissing-braces -Wnonnull -Wparentheses -Wpointer-sign -Wreorder -Wreturn-type -Wsequence-point -Wsign-compare** (only in C ++) **-Wstrict-aliasing -Wstrict-overflow=1 -Wswitch -Wtrigraphs -Wuninitialized -Wunknown-pragmas -Wunused-function -Wunused-label -Wunused-value -Wunused-variable -Wvolatile-register-var**

Note that some warning flags are not implied by **-Wall**. Some of them warn about constructions that users generally do not consider questionable, but which occasionally you might wish to check for; others warn about constructions that are necessary or hard to avoid in some cases, and there is no simple way to modify the code to suppress the warning. Some of them are enabled by **-Wextra** but many of them must be enabled individually.

Corrections

-Werror

Make all warnings into errors.

-Werror=

Make the specified warning into an error. The specifier for a warning is appended, for example **-Werror=switch** turns the warnings controlled by **-Wswitch** into errors. This switch takes a negative form, to be used to negate **-Werror** for specific warnings, for example **-Wno-error=switch** makes **-Wswitch** warnings not be errors, even when **-Werror** is in effect. You can use the **-fdiagnostics-show-option** option to have each controllable warning amended with the option which controls it, to determine what to use with this option.

Note that specifying **-Werror=***foo* automatically implies **-W***foo*. However, **-Wno-error=***foo* does not imply anything.

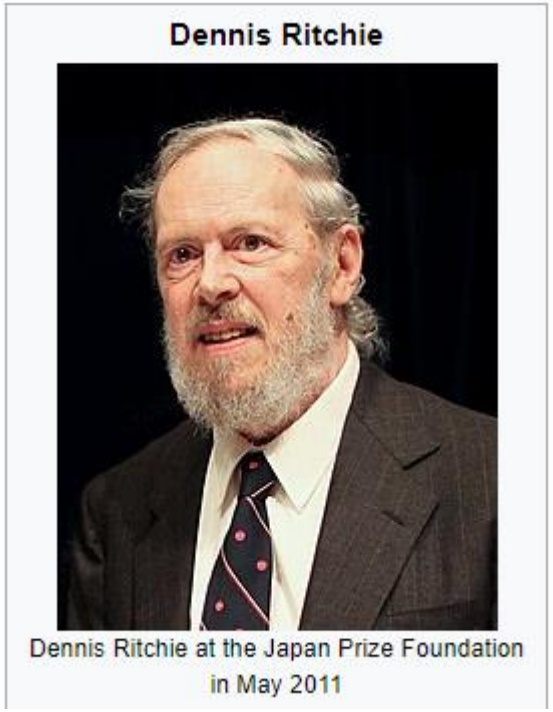
-Wfatal-errors

This option causes the compiler to abort compilation on the first error occurred rather than trying to keep going and printing further error messages.

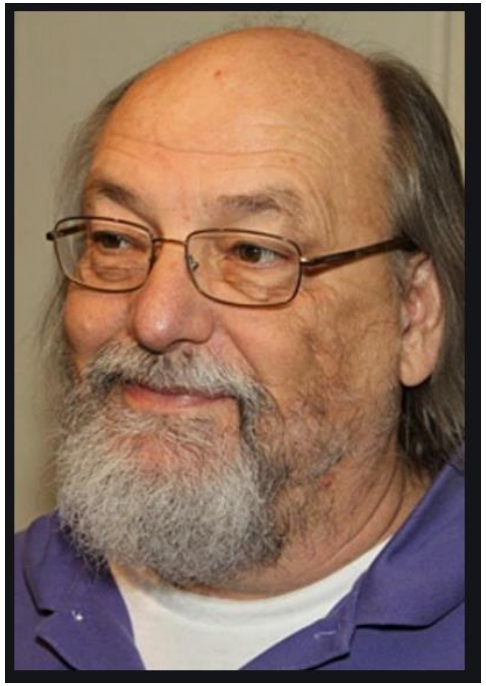
You can request many specific warnings with options beginning **-W**, for example **-Wimplicit** to request warnings on implicit declarations. Each of these specific warning options also has a negative form beginning

Java to C Demo

History



Ken Thompson



Bell Laboratories

1972

Unix Operating System

History

PDP-11/40. The processor is at the bottom. A TU56 dual DECtape drive is installed above it.

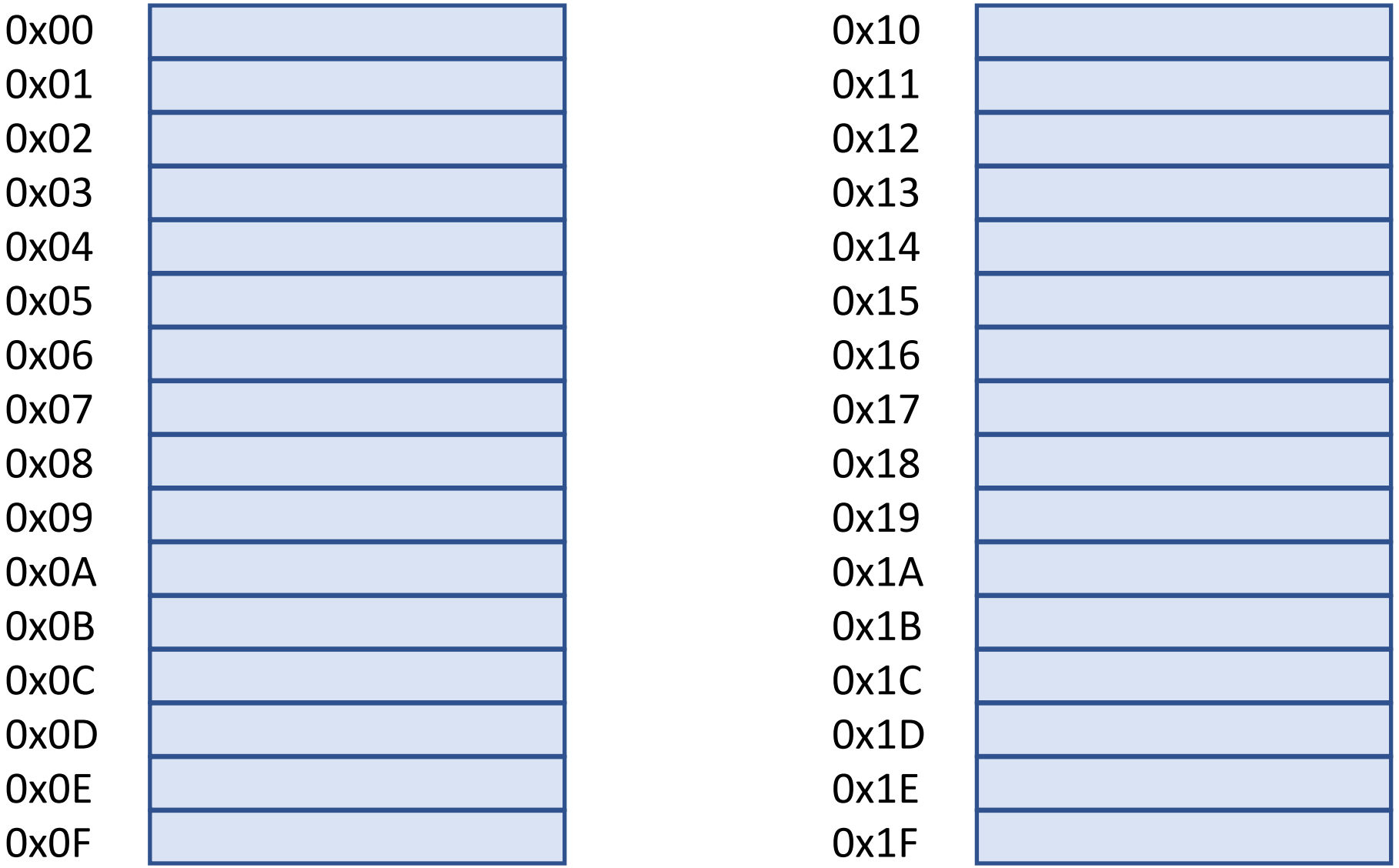
In 1972

16 bit machine with 128KB main memory
Compiler required 32 KB
20 simultaneous users connecting with dumb terminals
Connection speed 10 characters per second
3 lines per second compilation speed

VolumeOfCylinder = 1032 characters and 40 lines
approximately 2 minutes to communicate and compile the code

Efficiency and Speed!!!
No safety checks (variable assignment, array out of bounds)
Memory management and garbage collection
Space efficiency 16 character string = 17 bytes vs 76 in Java

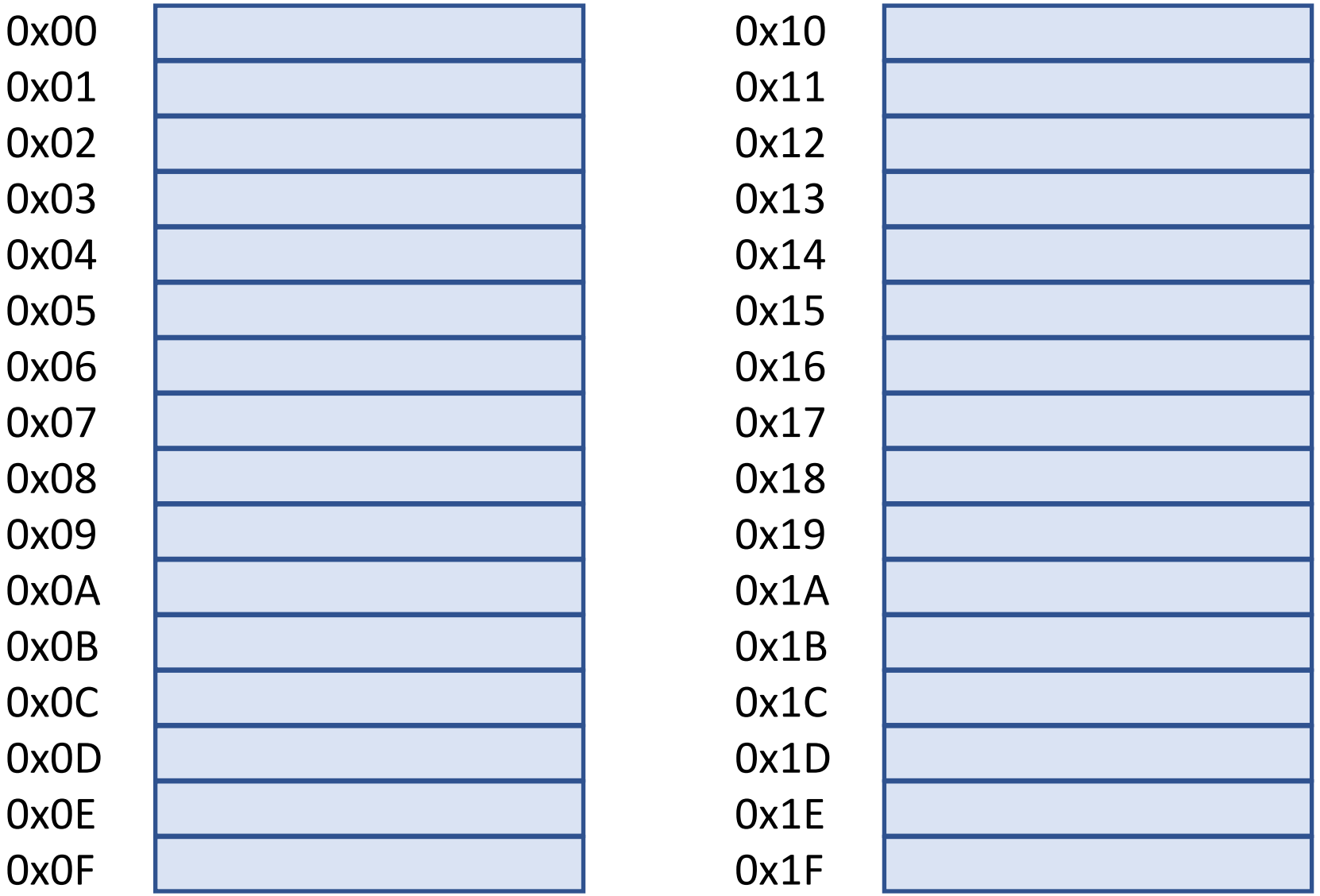
Memory Model



Memory Model

32 addresses
Require
$Log_2(32) = 5$ bits
So this would be a
5 bit machine

Min address = 00000
Max address = 11111

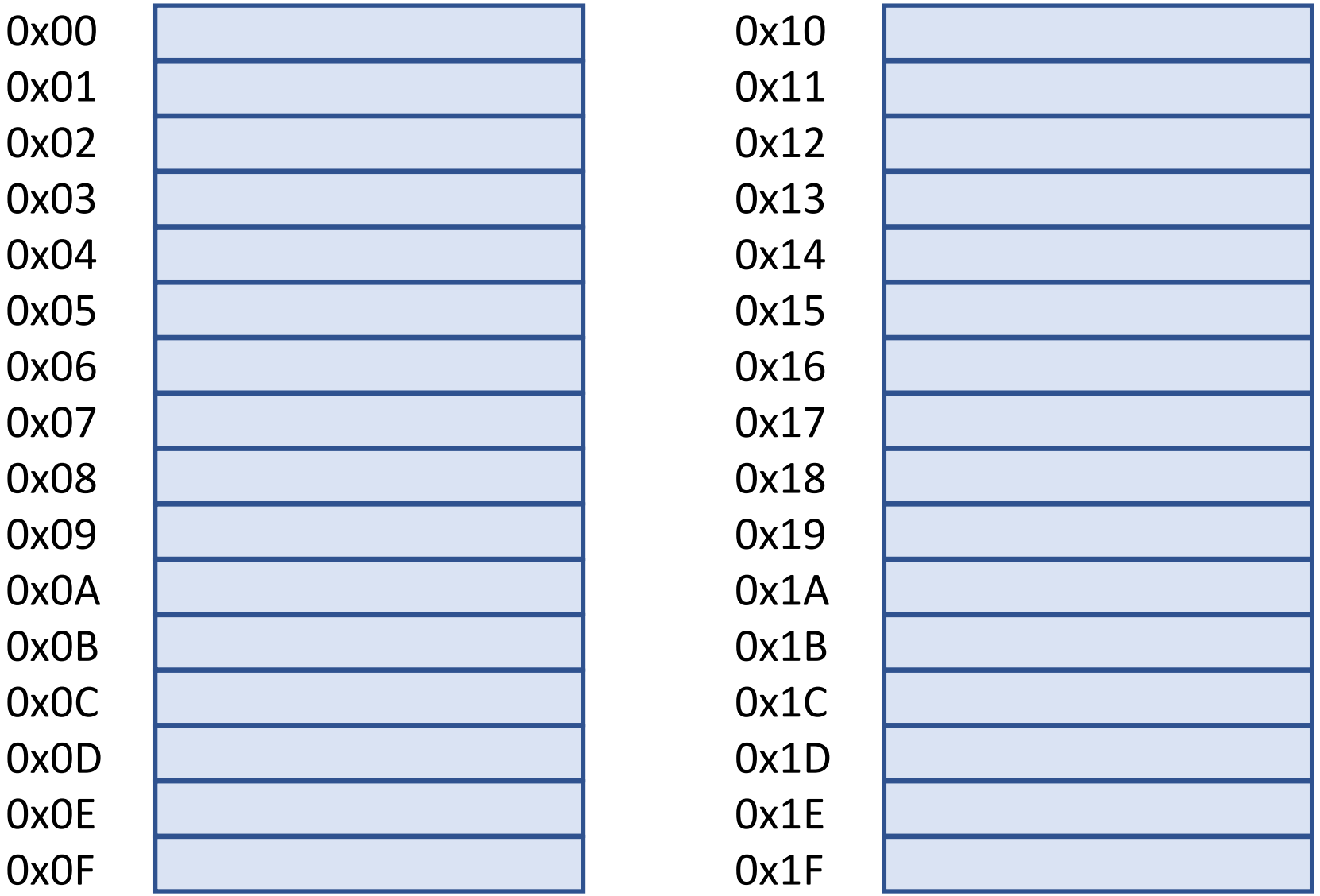


Memory Model

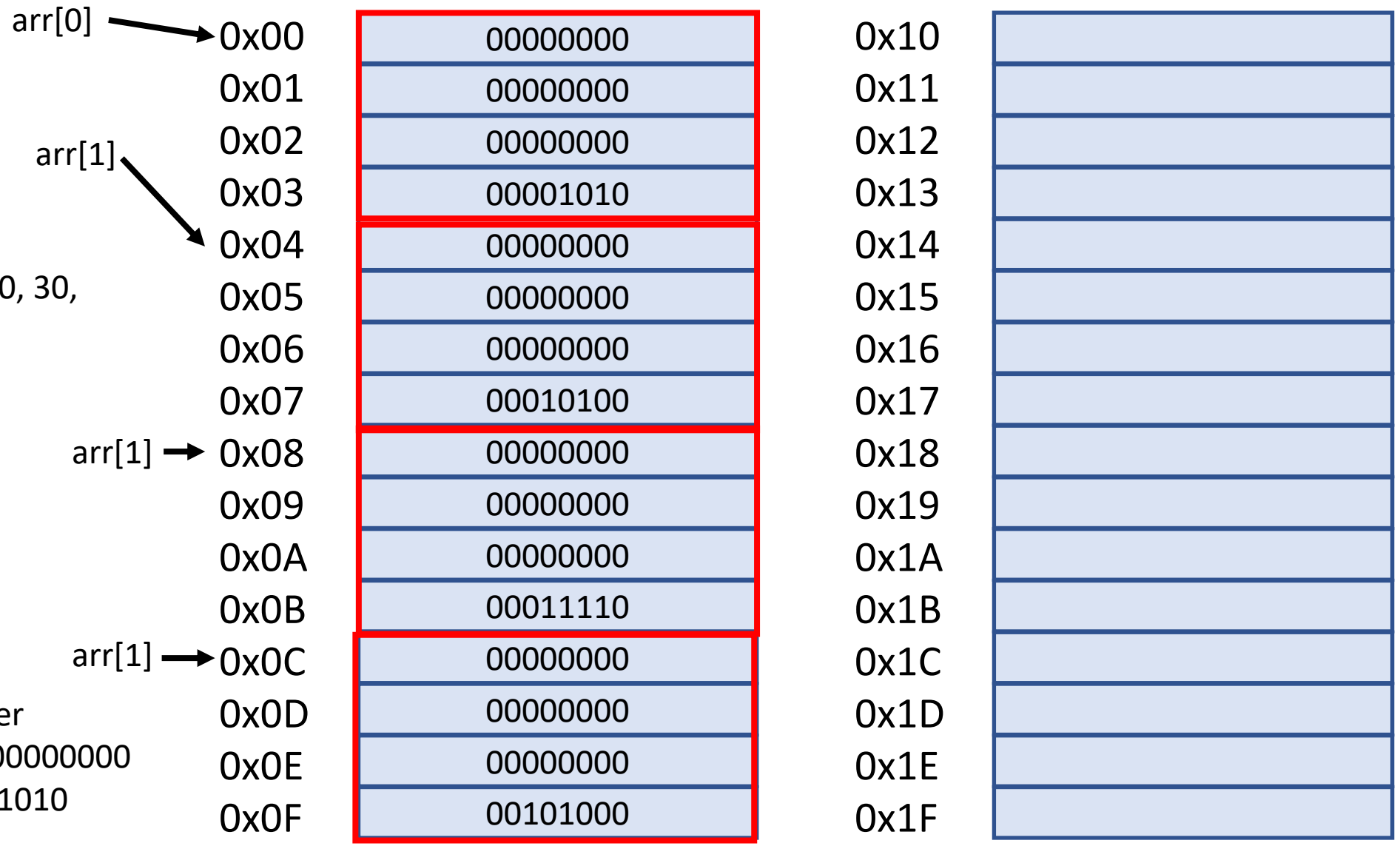
32 addresses
Require
$Log_2(32) = 5$ bits
So this would be a
5 bit machine

Min address = 00000
Max address = 11111

11111 -> 0001 1111
1 F



Memory Model



int arr[4] = {10, 20, 30, 40};

32 16 8 4 2 1

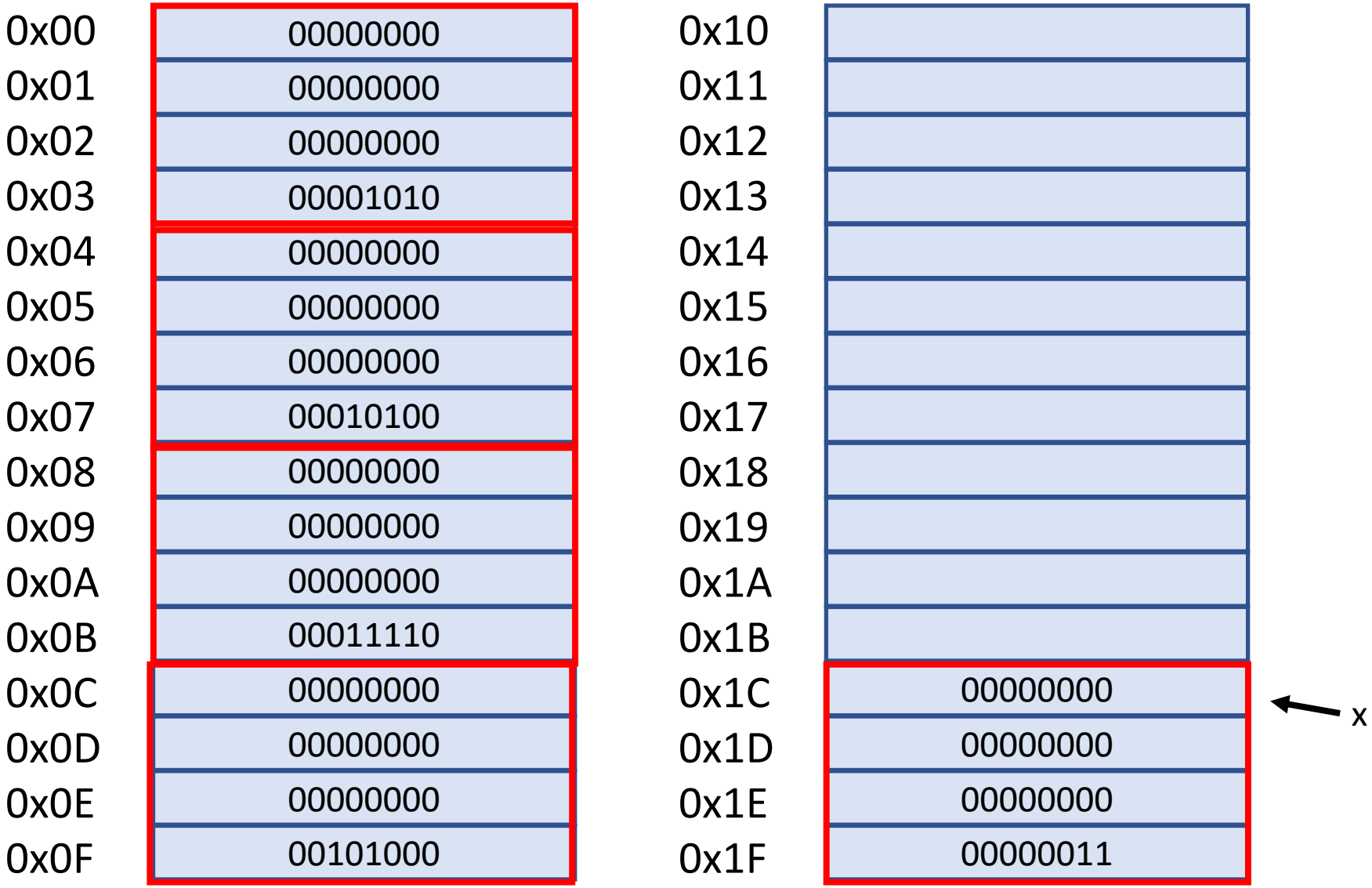
10 -> 001010
20 -> 010100
30 -> 011110
40 -> 101000

4 bytes per integer
10 -> 00000000 00000000 000000000 00001010

Memory Model

```
int x = 3;
printf("x = %d\n",x);
```

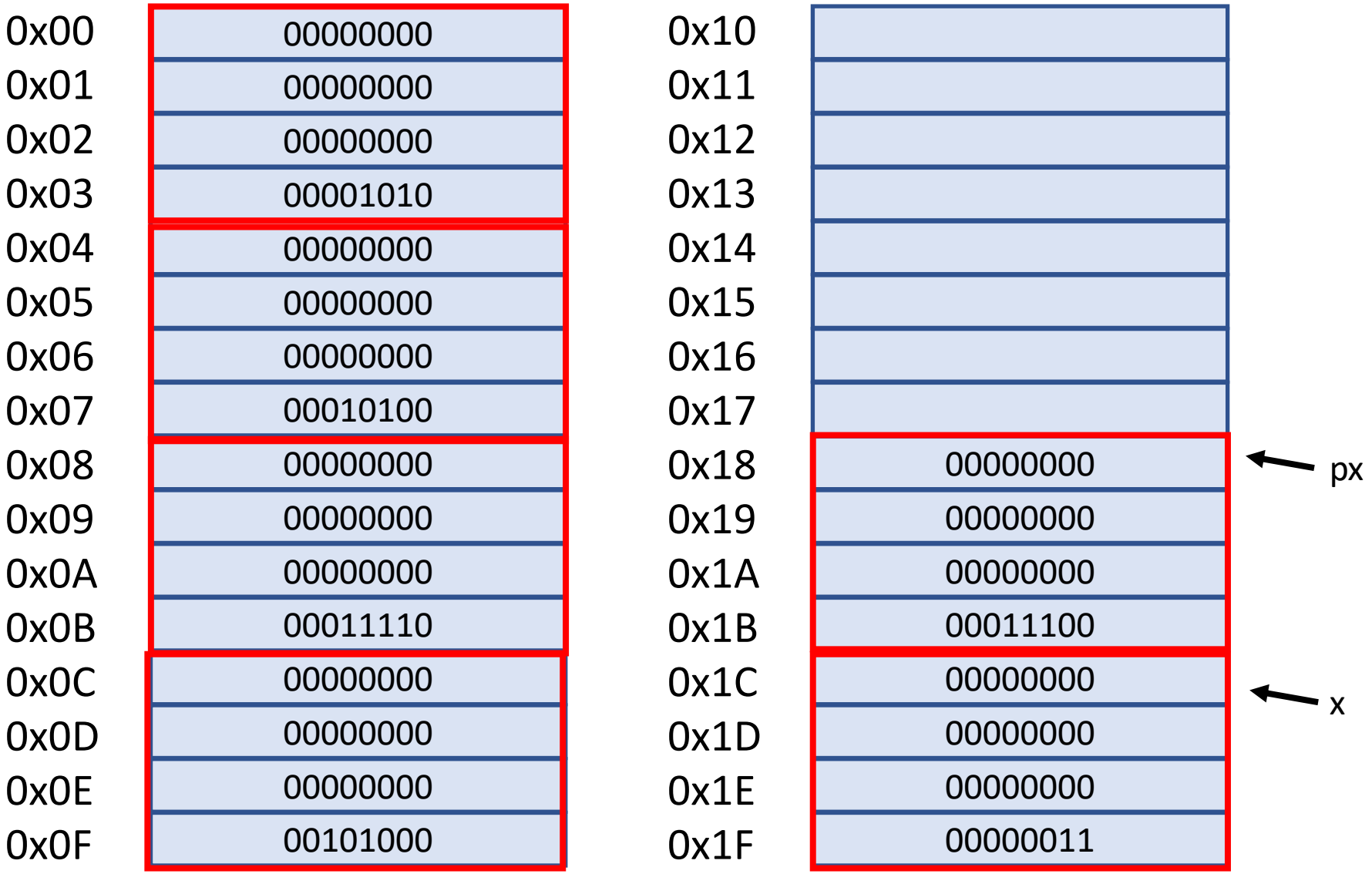
3



Memory Model

```
int x = 3;
int *px = &x
printf("x = %d\n",x);
printf("px = %o\n",px);
```

3
1C



Memory Model

```
int x = 3;
int *px = &x
printf("x = %d\n",x);
printf("px = %o\n",px);
printf("*px = %d\n",*px);
```

3
1C
3