

# Late G. N. Sapkal College of Engineering,

Anjaneri, Nashik-422212

## DEPARTMENT OF COMPUTER ENGINEERING



# LABORATORY PRACTICE-I MANUAL TE COMPUTER

Semester : V  
Academic Year: 2022-23

Practical Assessment: 25  
TW Assessment: 25

### Course Objectives:

- To learn system programming tools
- To learn modern operating system
- To learn various techniques, tools, applications in IoT and Embedded Systems /Human Computer Interface/Distributed Systems/ Software Project Management

### Course Outcomes:

- **Systems Programming and Operating System**
  - CO1: Implement language translators
  - CO2: Use tools like LEX and YACC
  - CO3: Implement internals and functionalities of Operating System
- **Software Project Management**
  - CO4: Apply Software Project Management tools
  - CO5: Implement software project planning and scheduling
  - CO6: Analyse staffing in software project

Prepared By,  
Prof. S. D. Bagade

## **GROUP-A**

### **ASSIGNMENT NO: 1.1**

#### **1. Title:**

Design suitable Data structures and implement Pass-I and Pass-II of a two-pass assembler for pseudo-machine. Implementation should consist of a few instructions from each category and few assembler directives. The output of Pass-I (intermediate code file and symbol table) should be input for Pass-II.

#### **2. Objectives:**

- To understand data structures to be used in pass I of an assembler.
- To implement pass I of an assembler

#### **3. Problem Statement:**

Write a program to create pass-I Assembler

#### **4. Outcomes:**

After completion of this assignment students will be able to:

- Understand the concept of Pass-I Assembler
- Understand the Programming language of Java

#### **5. Software Requirements:**

- Linux OS, JDK1.7

#### **6. Hardware Requirement:**

- 4GB RAM ,500GB HDD

#### **7. Theory Concepts:**

A language translator bridges an execution gap to machine language of computer system. An assembler is a language translator whose source language is assembly language.

An Assembler is a program that accepts as input an Assembly language program and converts it into machine language.

Language processing activity consists of two phases, Analysis phase and synthesis phase. Analysis of source program consists of three components, Lexical rules, syntax rules and semantic rules. Lexical rules govern the formation of valid statements in source language. Semantic rules associate the formation meaning with valid statements of language. Synthesis phase is concerned with construction of target language statements, which have the same meaning as source language statements. This consists of memory allocation and code generation.

## **TWO PASS TRANSLATION SCHEME:**

In a 2-pass assembler, the first pass constructs an intermediate representation of the source program for use by the second pass. This representation consists of two main components - data structures like Symbol table, Literal table and processed form of the source program called as intermediate code(IC). This intermediate code is represented by the syntax of Variant -I.

Analysis of source program statements may not be immediately followed by synthesis of equivalent target statements. This is due to forward references issue concerning memory requirements and organization of Language Processor (LP).

Forward reference of a program entity is a reference to the entity, which precedes its definition in the program. While processing a statement containing a forward reference, language processor does not possess all relevant information concerning referenced entity. This creates difficulties in synthesizing the equivalent target statements. This problem can be solved by postponing the generation of target code until more information concerning the entity is available. This also reduces memory requirements of LP and simplifies its organization. This leads to multi-pass model of language processing.

### ***Language Processor Pass: -***

It is the processing of every statement in a source program or its equivalent representation to perform language-processing function.

### ***Assembly Language statements: -***

There are three types of statements Imperative, Declarative, Assembly directives. An imperative statement indicates an action to be performed during the execution of assembled program. Each imperative statement usually translates into one machine instruction. Declarative statement e.g. DS reserves areas of memory and associates names with them. DC constructs memory word containing constants. Assembler directives instruct the assembler to perform certain actions during assembly of a program,

e.g. START<constant> directive indicates that the first word of the target program generated by assembler should be placed at memory word with address <constant>

## **Function Of Analysis And Synthesis Phase:**

### ***Analysis Phase: -***

Isolate the label operation code and operand fields of a statement.

Enter the symbol found in label field (if any) and address of next available machine word into symbol table.

Validate the mnemonic operation code by looking it up in the mnemonics table. Determine the machine storage requirements of the statement by considering the mnemonic operation code and operand fields of the statement.

Calculate the address of the address of the first machine word following the target code generated for this statement (Location Counter Processing)

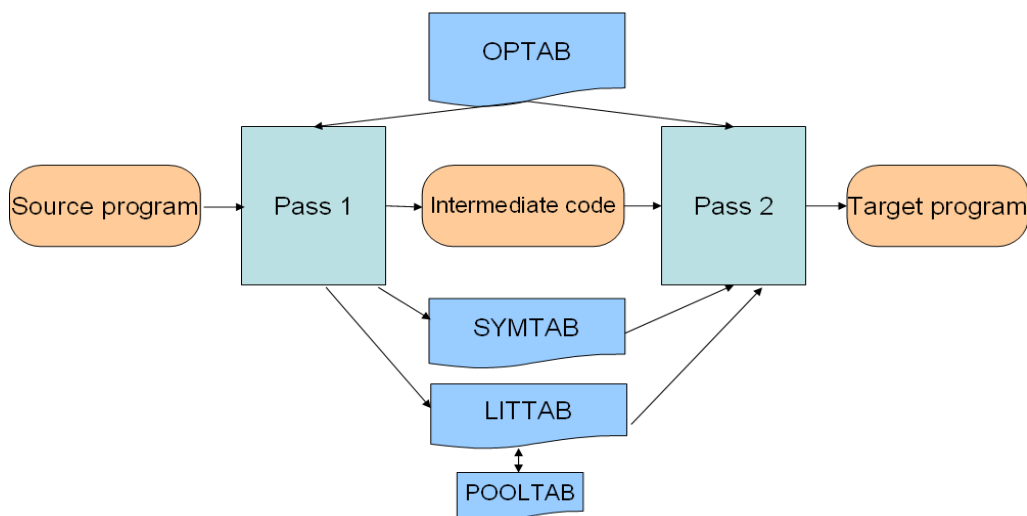
### **Synthesis Phase:**

Obtain the machine operation code corresponding to the mnemonic operation code by searching the mnemonic table.

Obtain the address of the operand from the symbol table.

Synthesize the machine instruction or the machine form of the constant as the case may be.

### **DATA STRUCTURES OF A TWO PASS ASSEMBLER:**



Data Structure of Assembler:

a) Operation code table (OPTAB) :This is used for storing mnemonic, operation code and class of instruction

Structure of OPTAB is as follows

b) Data structure updated during translation: Also called as translation time data

structure. They are

I. SYMBOL TABLE (SYMTAB) : It contains entries such as symbol, its address and value.

**SYMBOL TABLE** have following fields :

Name of symbol	Symbol Address	Value
----------------	----------------	-------

II. LITERAL TABLE (LITAB) : it contains entries such as literal and its value.

**Literal Table** has following fields :

literal	Address of Literal
---------	--------------------

III . POOL TABLE (POOLTAB): Contains literal number of the starting literal of each literal pool.

**Pool TABLE** (pooltab) have following fields.

LITERAL_NO

IV: Location Counter which contains address of next instruction by calculating length of each instruction.

**Design of a Two Pass Assembler: -**

Tasks performed by the passes of two-pass assembler are as follows:

**Pass I: -**

Separate the symbol, mnemonic opcode and operand fields.

Determine the storage-required for every assembly language statement and update the location counter.

Build the symbol table and the literal table.

Construct the intermediate code for every assembly language statement.

**Pass II: -**

Synthesize the target code by processing the intermediate code generated during pass1

**INTERMEDIATE CODE REPRESENTATION**

The intermediate code consists of a set of IC units, each IC unit consisting of the following three fields

1. Address
2. Representation of the mnemonic opcode
3. Representation of operands

**Mnemonic field**

The mnemonic field contains a pair of the form: (*statement class, code*)

Where *statement class* can be one of IS,DL and AD standing for imperative statement, declaration statement and assembler directive , respectively.

For imperative statement, *code* is the instruction opcode in the machine language

For declaration and assembler directives , following are the codes

Declaration Statements

DC    01  
DS    02

Assembler directives

START    01  
END       02  
ORIGIN    03  
EQU       04  
LTORG     05

**Mnemonic Operation Codes :**

Instruction Opcode	Assembly Mnemonic	Remarks
<b>00</b>	<b>STOP</b>	<b>STOP EXECUTION</b>
<b>01</b>	<b>ADD</b>	<b>FIRST OPERAND IS MODIFIED CONDITION CODE IS SET</b>
<b>02</b>	<b>SUB</b>	<b>FIRST OPERAND IS MODIFIED CONDITION</b>

		<b>CODE IS SET</b>
<b>03</b>	<b>MULT</b>	<b>FIRST OPERAND IS MODIFIED CONDITION CODE IS SET</b>
<b>04</b>	<b>MOVER</b>	<b>REGISTER ← MEMORY MOVE</b>
<b>05</b>	<b>MOVEM</b>	<b>MEMORY MOVE → REGISTER MOVE</b>
<b>06</b>	<b>COMP</b>	<b>SETS CONDITION CODE</b>
<b>07</b>	<b>BC</b>	<b>BRANCH ON CONDITION</b>
<b>08</b>	<b>DIV</b>	<b>ANALOGOUS TO SUB</b>
<b>09</b>	<b>READ</b>	<b>FIRST OPERAND IS NOT USED</b>
<b>10</b>	<b>PRINT</b>	<b>FIRST OPERAND IS NOT USED</b>

### Branch On Condition :

BC     <condition code >     <memory address>

Transfer Control to the Memory word With Address < memory address>

<b>Condition code</b>	<b>Opcode</b>
LT	01
LE	02
EQ	03
GT	04
GE	05
ANY	06

### 8. Algorithms(procedure) :

#### PASS 1

- Initialize location counter, entries of all tables as zero.
- Read statements from input file one by one.
- While next statement is not END statement

I. Tokenize or separate out input statement as label,numonic,operand1,operand2

II. If label is present insert label into symbol table.

- III. If the statement is LTORG statement processes it by making it's entry into literal table, pool table and allocate memory.
- IV. If statement is START or ORIGIN Process location counter accordingly.
- V. If an EQU statement, assign value to symbol by correcting entry in symbol table.
- VI. For declarative statement update code, size and location counter.
- VII. Generate intermediate code.
- VIII. Pass this intermediate code to pass -2.

### **10. Conclusion:**

Thus, I have studied visual programming and implemented dynamic link library application for arithmetic operation

### **Questions: [Write short answer]**

1. Explain what is meant by pass of an assembler.
2. Explain the need for two pass assembler.
3. Explain terms such as Forward Reference and backward reference.
4. Explain various types of errors that are handled in pass-I.
5. Explain the need of Intermediate Code generation and the variants used.
6. State various tables used and their significance in the design of two pass Assembler.
7. What is the job of assembler?
8. What are the various data structures used for implementing Pass-I of a two-pass assembler.
9. What feature of assembly language makes it mandatory to design a two pass assembler?
10. How are literals handled in an assembler?
11. How assembler directives are handled in pass I of assembler?



## ASSIGNMENT NO: 1.2

### 1. Title:

Implement Pass-II of two pass assembler for pseudo-machine in Java using object oriented features. The output of assignment-1 (intermediate file and symbol table) should be input for this assignment.

### 2. Objectives:

- To understand data structures to be used in pass II of an assembler.
- To implement pass I of an assembler

### 3. Problem Statement:

Write a program to create pass-II Assembler

### 4. Outcomes:

After completion of this assignment students will be able to:

- Understand the concept of Pass-II Assembler
- Understand the Programming language of Java

### 5. Software Requirements:

- Linux OS, JDK1.7

### 6. Hardware Requirement:

- 4GB RAM ,500GB HDD

### 7. Theory Concepts:

#### Design of a Two Pass Assembler: -

Tasks performed by the passes of two-pass assembler are as follows:

#### *Pass I: -*

Separate the symbol, mnemonic opcode and operand fields.

Determine the storage-required for every assembly language statement and update the location counter.

Build the symbol table and the literal table.

Construct the intermediate code for every assembly language statement.

***Pass II: -***

Synthesize the target code by processing the intermediate code generated during pass1

**Data Structure used by Pass II:**

1. OPTAB: A table of mnemonic opcodes and related information.
2. SYMTAB: The symbol table
3. POOL\_TAB and LITTAB: A table of literals used in the program
4. Intermediate code generated by Pass I
5. Output file containing Target code / error listing.

**8. Algorithms(procedure) :**

**Algorithm :**

1. code\_area\_address=address of code area;

Pooltab\_ptr:=1;

loc\_cntr=0;

2. While next statement is not an END statement

a) clear the machine\_code\_buffer

b) if an LTORG statement

I) process literals in LITTAB[POOLTAB[pooltab\_ptr]]...

LITTAB[POOLTAB[pooltab\_ptr+1]]-1 similar to processing of constants in a dc statement.

II) size=size of memory area required for literals

III) pooltab\_ptr=pooltab\_ptr+1

c) if a START or ORIGIN statement then

I) loc\_cntr = value specified in operand field

II) size=0;

d) if a declaration statement

I) if a DC statement then assemble the constant in machine\_code\_buffer

II) size=size of memory area required by DC or DS:

e) if an imperative statement then

I) get operand address from SYMTAB or LITTAB

II) Assemble instruction in machine code buffer.

III) size=size of instruction;

f) if size  $\neq$  0 then

I) move contents of machine\_code\_buffer to the address code\_area\_address+loc\_cntr ;

II) loc\_cntr=loc\_cntr+size;

3. (Processing of END statement)

### **9. Conclusion:**

Thus, I have studied visual programming and implemented dynamic link library application for arithmetic operation

### **Questions: [Write short answer]**

1. Explain various types of errors that are handled in pass-II.
2. Write algorithm of pass-II.
3. Draw flowchart of pass-II.
4. State various tables used and their significance in the design of two pass Assembler.
5. How LTORG statement is handled in pass II of assembler?
6. How Declarative statement is handled in pass II of assembler?
7. What is the significance of pool table?
8. Which data structures of pass I are used in pass II of assembler?
9. Explain the handling of imperative statement.
10. What feature of assembly language makes it mandatory to design a two pass assembler?
11. How are literals handled in an assembler?
12. How assembler directives are handled in pass I of assembler?

## ASSIGNMENT NO: 02

### 1. Title:

Write a program to create Dynamic Link Library for any mathematical operation and write an application program to test it. (Java Native Interface / Use VB or VC++).

### 2. Objectives:

- To understand Dynamic Link Libraries Concepts
- To implement dynamic link library concepts
- To study about Visual Basic

### 3. Problem Statement:

Write a program to create Dynamic Link Library for Arithmetic Operation in VB.net

### 4. Outcomes:

After completion of this assignment students will be able to:

- Understand the concept of Dynamic Link Library
- Understand the Programming language of Visual basic

### 5. Software Requirements:

- Visual Studio 2010

### 6. Hardware Requirement:

- M/C Lenovo Think center M700 Ci3,6100,6th Gen. H81, 4GB RAM ,500GB HDD

### 7. Theory Concepts:

#### Dynamic Link Library:

A dynamic link library (DLL) is a collection of small programs that can be loaded when needed by larger programs and used at the same time. The small program lets the larger program communicate with a specific device, such as a printer or scanner. It is often packaged as a DLL program, which is usually referred to as a DLL file. DLL files that support specific device operation are known as device drivers.

A DLL file is often given a ".dll" file name suffix. DLL files are dynamically linked with the program that uses them during program execution rather than being compiled into the main program.

The advantage of DLL files is space is saved in random access memory (RAM) because the files don't get loaded into RAM together with the main program. When a DLL file is needed, it is loaded and run. For example, as long as a user is editing a document in Microsoft Word, the printer DLL file does not need to be loaded into RAM. If the user decides to print the document, the Word application causes the printer DLL file to be loaded and run.

A program is separated into modules when using a DLL. With modularized components, a program can be sold by module, have faster load times and be updated without altering other parts of the program. DLLs help operating systems and programs run faster, use memory efficiently and take up less disk space.

### **Feature of DLL:**

- DLLs are essentially the same as EXEs, the choice of which to produce as part of the linking process is for clarity, since it is possible to export functions and data from either.
- It is not possible to directly execute a DLL, since it requires an EXE for the operating system to load it through an entry point, hence the existence of utilities like RUNDLL.EXE or RUNDLL32.EXE which provide the entry point and minimal framework for DLLs that contain enough functionality to execute without much support.
- DLLs provide a mechanism for shared code and data, allowing a developer of shared code/data to upgrade functionality without requiring applications to be re-linked or re-compiled. From the application development point of view Windows and OS/2 can be thought of as a collection of DLLs that are upgraded, allowing applications for one version of the OS to work in a later one, provided that the OS vendor has ensured that the interfaces and functionality are compatible.
- DLLs execute in the memory space of the calling process and with the same access permissions which means there is little overhead in their use but also that there is no protection for the calling EXE if the DLL has any sort of bug.

### **Difference between the Application & DLL:**

- An application can have multiple instances of itself running in the system simultaneously, Whereas a DLL can have only one instance.
- An application can own things such as a stack, global memory, file handles, and a message queue, but a DLL cannot.

### **Executable file links to DLL:**

An executable file links to (or loads) a DLL in one of two ways:

- Implicit linking
- Explicit linking

Implicit linking is sometimes referred to as static load or load-time dynamic linking. Explicit linking is sometimes referred to as dynamic load or run-time dynamic linking.

With implicit linking, the executable using the DLL links to an import library (.lib file) provided by the maker of the DLL. The operating system loads the DLL when the executable using it is loaded. The client executable calls the DLL's exported functions just as if the functions were contained within the executable.

With explicit linking, the executable using the DLL must make function calls to explicitly load and unload the DLL and to access the DLL's exported functions. The client executable must call the exported functions through a function pointer.

An executable can use the same DLL with either linking method. Furthermore, these mechanisms are not mutually exclusive, as one executable can implicitly link to a DLL and another can attach to it explicitly.

## Calling DLL function from Visual Basic Application:

For Visual Basic applications (or applications in other languages such as Pascal or Fortran) to call functions in a C/C++ DLL, the functions must be exported using the correct calling convention without any name decoration done by the compiler.

stdcall creates the correct calling convention for the function (the called function cleans up the stack and parameters are passed from right to left) but decorates the function name differently. So, when **declspec(dllexport)** is used on an exported function in a DLL, the decorated name is exported.

The\_stdcall name decoration prefixes the symbol name with an underscore (\_) and appends the symbol with an at sign (@) character followed by the number of bytes in the argument list (the required stack space). As a result, the function when declared as:

```
int_stdcall func (int a, double b)
```

is decorated as:

```
_func@12
```

The C calling convention (\_cdecl) decorates the name as \_func.

To get the decorated name, use [/MAP](#). Use of **declspec(dllexport)** does the following:

- If the function is exported with the C calling convention (**\_cdecl**), it strips the leading underscore (\_) when the name is exported.
- If the function being exported does not use the C calling convention (for example, stdcall), it exports the decorated name.

Because there is no way to override where the stack cleanup occurs, you must use\_stdcall. To undecorate names with\_stdcall, you must specify them by using aliases in the EXPORTS section of the .def file. This is shown as follows for the following function declaration:

```
int_stdcall MyFunc (int a, double b);  
void_stdcall InitCode (void);
```

In the .DEF file:

```
EXPORTS  
MYFUNC=_MyFunc@12  
INITCODE=_InitCode@0
```

For DLLs to be called by programs written in Visual Basic, the alias technique shown in this topic is needed in the .def file. If the alias is done in the Visual Basic program, use of aliasing in the .def file is not necessary. It can be done in the Visual Basic program by adding an alias clause to the [Declare](#) statement.

### **DLL's Advantages:**

- Saves memory and reduces swapping. Many processes can use a single DLL simultaneously, sharing a single copy of the DLL in memory. In contrast, Windows must load a copy of the library code into memory for each application that is built with a static link library.
- Saves disk space. Many applications can share a single copy of the DLL on disk. In contrast, each application built with a static link library has the library code linked into its executable image as a separate copy.
- Upgrades to the DLL are easier. When the functions in a DLL change, the applications that use them do not need to be recompiled or relinked as long as the function arguments and return values do not change. In contrast, statically linked object code requires that the application be relinked when the functions change.
- Provides after-market support. For example, a display driver DLL can be modified to support a display that was not available when the application was shipped.
- Supports multi language programs. Programs written in different programming languages can call the same DLL function as long as the programs follow the function's calling convention. The programs and the DLL function must be compatible in the following ways: the order in which the function expects its arguments to be pushed onto the stack, whether the function or the application is responsible for cleaning up the stack, and whether any arguments are passed in registers.
- Provides a mechanism to extend the MFC library classes. You can derive classes from the existing MFC classes and place them in an MFC extension DLL for use by MFC applications.
- Eases the creation of international versions. By placing resources in a DLL, it is much easier to create international versions of an application. You can place the strings for each language version of your application in a separate resource DLL and have the different language versions load the appropriate resources.

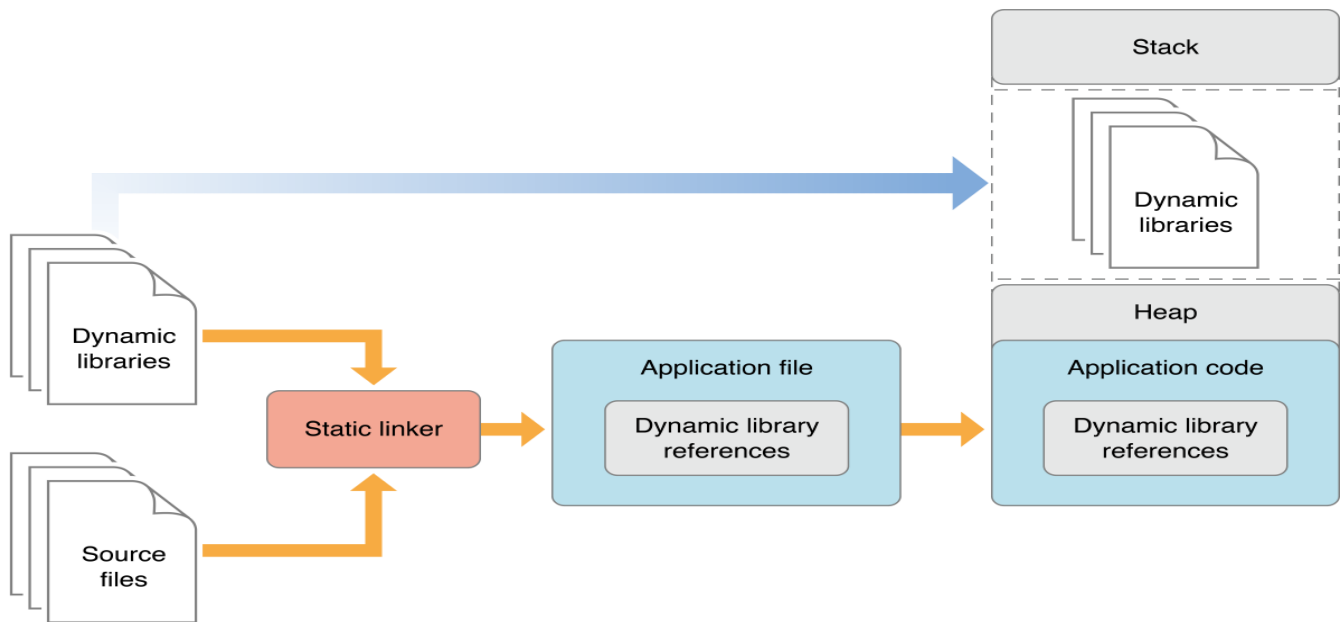
### **Disadvantage:**

- A potential disadvantage to using DLLs is that the application is not self-contained; it depends on the existence of a separate DLL module.

### **Visual Basic:**

Visual Basic is a third-generation event-driven programming language first released by Microsoft in 1991. It evolved from the earlier DOS version called BASIC. **BASIC** means **B**eginners' **A**ll- purpose **S**ymbolic **I**nstruction **C**ode. Since then Microsoft has released many versions of Visual Basic, from Visual Basic 1.0 to the final version Visual Basic 6.0. Visual Basic is a user-friendly programming language designed for beginners, and it enables anyone to develop GUI window applications easily. In 2002, Microsoft released Visual Basic.NET (VB.NET) to replace Visual Basic 6. Thereafter, Microsoft declared VB6 a legacy programming language in 2008. Fortunately, Microsoft still provides some form of support for VB6. VB.NET is a fully object-oriented programming language implemented in the .NET Framework. It was created to cater for the development of the web as well as mobile applications. However, many developers still favor Visual Basic 6.0 over its successor Visual Basic.NET.

## 8. Design (architecture):



## 9. Conclusion:

Thus, I have studied visual programming and implemented dynamic link library application for arithmetic operation

## References:

[https://en.wikipedia.org/wiki/Dynamic-link\\_library](https://en.wikipedia.org/wiki/Dynamic-link_library)

[https://en.wikipedia.org/wiki/Visual\\_Basic](https://en.wikipedia.org/wiki/Visual_Basic)

[https://www.google.co.in/search?q=dynamic+link+library+architecture&dcr=0&source=lnms&tbn=isch&sa=X&ved=0ahUKEwjgubTAuJvZAhWHO48KHRZbD7sO\\_AUICigB&biw=1366&bih=651#imgre=LU8YqljE8-afxM](https://www.google.co.in/search?q=dynamic+link+library+architecture&dcr=0&source=lnms&tbn=isch&sa=X&ved=0ahUKEwjgubTAuJvZAhWHO48KHRZbD7sO_AUICigB&biw=1366&bih=651#imgre=LU8YqljE8-afxM)

<https://msdn.microsoft.com/en-us/library/9vd93633.aspx>



**Questions: [Write short answer]**

1. What Is Dll And What Are Their Usages And Advantages?
2. What Are The Sections In A Dll Executable/binary?
3. Where Should We Store Dlls ?
4. Who Loads And Links The Dlls?
5. How Many Types Of Linking Are There?
6. What Is Implicit And Explicit Linking In Dynamic Loading?
7. How to call a DLL function from Visual Basic?

## ASSIGNMENT NO: 3.1

### 1. Title:

Design suitable data structures and implement pass-I of a two-pass macro-processor. The output of Pass-I (MNT, MDT and intermediate code file without any macro definitions) should be input for Pass-II.

### 2. Objectives:

- To understand Data structure of Pass-1 macro processor
- To understand Pass-1 macro processor concept
- To understand macro facility.

### 3. Problem Statement:

Design suitable data structures and implement pass-I of a two-pass macro-processor.

### 4. Outcomes:

After completion of this assignment students will be able to:

- Implemented Pass – 1 macro processor
- Implemented MNT, MDT table.
- Understood concept Pass-1 macro processor.

### 5. Software Requirements:

Latest JDK, Eclipse

### 6. Hardware Requirement:

- M/C Lenovo Think center M700 Ci3,6100,6th Gen. H81, 4GB RAM ,500GB HDD

### 7. Theory Concepts:

#### Macro Processor

Macro pre-processor takes a source program containing macro definitions and macro calls and translates into an assembly language program without any macro definitions or calls. This program can now be handed over to a conventional assembler to obtain the target language (as shown in Fig. 2.5.1).



(S3.6)Fig. 2.5.1 : A scheme for a macro pre-processor

**Macro:**

Macro allows a sequence of source language code to be defined once & then referred to by name each time it is referred. Each time this name occurs in a program , the sequence of codes is substituted at that point.

Macro has following parts:-

- (1) Name of macro
- (2) Parameters in macro
- (3) Macro Definition

Parameters are optional.

**How To Define a Macro:-**

Macro can be formatted in following order:-

Start of definition	MACRO
macro name	mymacro
macro body	[ ADD AREG, X ADD BREG, X
End of macro definition	MEND

‘MACRO’ pseudo-op is the first line of definition & identifies the following line as macro instruction name.

Following the name line is sequence of instructions being abbreviated the instructions comprising the ‘MACRO’ instruction.

The definition is terminated by a line with MEND pseudo-op.

**Example of Macro:-**

- (1) Macro without parameters

MACRO

my macro

ADD AREG,X

ADD BREG,X

MEND

(2) Macro with parameters

MACRO

add macro &A

ADD AREG,

&A ADD

BREG,&A

MEND

The macro parameters (Formal Parameters) are initialized with '&'. Used as it is in operation. Formal Parameters are those which are in definition of macro.

Whereas while calling macros we use Actual Parameters.

### **How to Call a Macro?**

A macro is called by writing macro name with actual parameters in an Assembly Program.

Macro call leads to Macro Expansion.

Syntax: <macro-name> [<list of parameters>]

Example: - for above definitions of macro...

(1) mymacro

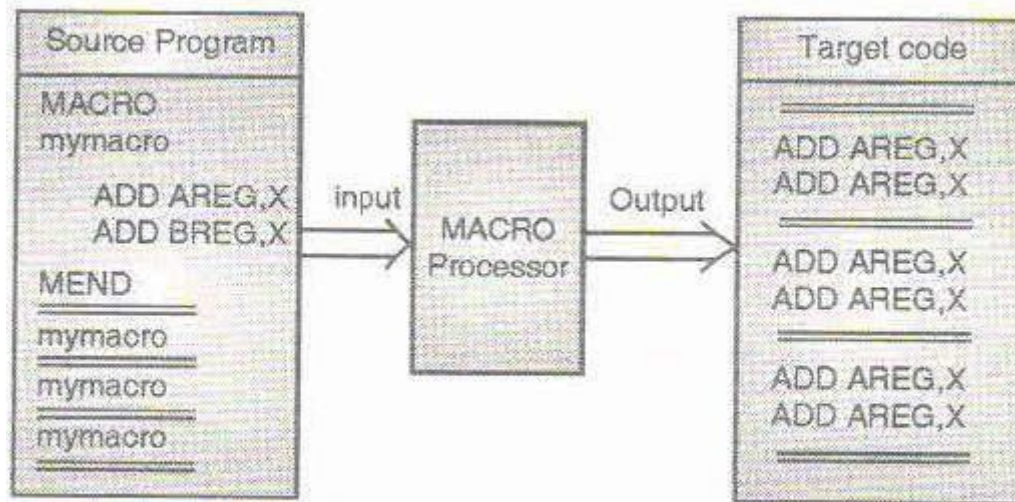
(2) addmacro X

### **Macro Expansion:-**

Each Call to macro is replaced by its body.

During Replacement, actual parameter is used in place of formal parameter.

- During Macro expansion, each statement forming the body of macro is picked up one by one sequentially.
- Each Statement inside the macro may have:
  - (1) An ordinary string, which is copied as it is during expansion.
  - (2) The name of a formal parameter which is proceeded by character '&'.
- During macro expansion an ordinary string is retained without any modification. Formal Parameters (Strings starting with &) is replaced by the actual parameter value.



(S3.2) Fig. 2.1.2 : Macro expansion

### Macro with Keyword Parameters:-

These are the methods to call macros with formal parameters. These formal parameters are of two types

#### (1) Positional Parameters:

Initiated with '&'.

Ex:- my macro &X

#### (2) Keyword Parameters:

Initiated with '&'. But has some default value.

During a call to macro, a keyword parameter is specified by its name.

Ex:- mymacro &X=A

### Nested Macro Calls:-

Nested Macro Calls are just like nested function calls in our normal calls. Only the transfer of control from one macro to other is done.

Consider this example:-

```

MACRO
  Innermacro
    ADD AREG,X
  MEND
  
```

```

MACRO
  outermacro
    innermacro
  
```

ADD AREG, Y  
MEND

Outer macro

In this example, firstly the MACRO outer macro gets executed & then inner macro.  
So Output will be Adding X & Y values in AREG register.

**Algorithm:**

Scan all MACRO definition one by one.

- (a) Enter its name in macro name table (MNT).
- (b) Store the entire macro definition in the macro definition table (MDT).
- (c) Add the information in the MNT indicates where definition of macro can be found in MDT.
- (d) Prepare argument list array (ALA).

**Data Structures of Two Pass Macros:**

1] Macro Name Table Pointer (MNTP) :

2] Macro Definition Table Pointer (MDTP) :

3] Macro Name Table :

- macro number(i.e pointer referenced from MNTP)
- Name of macros
- MDTP (i.e points to start position to MDT)

4] Macro Definition Table :

- Location Counter(where MDTP points to start position of macro)
- Opcode
- Rest (i.e it will contain the other part than opcodes used in macro).

5] Argument List Array :

- Index given to parameter
- Name of parameter

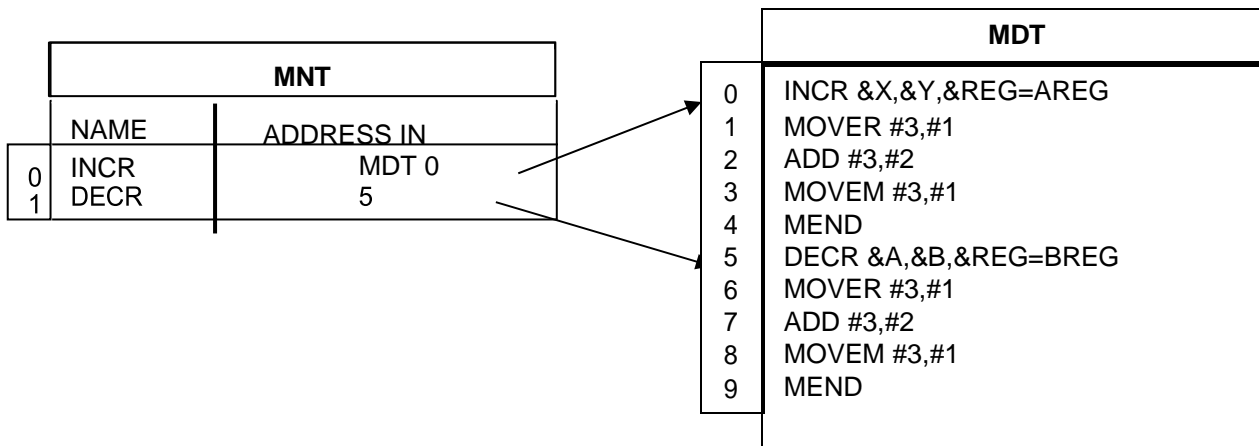
### Example for pass I of macro processor: -

#### Assembly Code Segment:-

```
MACRO
INCR &X,

Y,&REG=AREG MOVER
&REG,&Y
ADD &REG, &Y MOVEM
&REG, &X MEND
MACRO
DECR &A,&B,&REG=BREG
MOVER &REG,&A
ADD &REG,&B
MOVEM &REG,&A
MEND
START
READ N1
READ N2
INCR N1,N2,REG=CREG
DECR N1,N2
STOP
N1 DS 1
N2 DS 2
END
```

Output of Pass-I of Macro Assembler:-



**#1 – First Parameter**

**#2 – Second Parameter**

**#3 – Third Parameter**

## **8. Conclusion :**

Write your own conclusion.

### **Questions: [Write short answer]**

1. Compare assembler and macroprocessor
2. What is macro, macro expansion.
3. How to expand macro.



## **ASSIGNMENT NO - 3.2**

### **1. Title:**

Design suitable data structures and implement Pass-II of a two-pass macro-processor. The output of Pass-I (MNT, MDT and intermediate code file without any macro definitions) should be input for Pass-II.

### **2. Objectives:**

- To understand Data structure Pass-2 macro processor
- To understand Pass-1 & Pass-2 macro processor concept
- To understand Advanced macro facility

### **3. Problem Statement:**

Write a Java program for pass-II of a two-pass macro-processor. The output Pass-I (MNT, MDT and file without any macro definitions) should be input for this assignment

### **4. Outcomes:**

After completion of this assignment students will be able to:

- Implemented Pass – 2 macro processor
- Implemented machine code from MDT and MNT table.
- Understood concept Pass-2 macro processor.

### **5. Software Requirements:**

Latest JDK, Eclipse

### **6. Hardware Requirement:**

- M/C Lenovo Think center M700 Ci3,6100,6th Gen. H81, 4GB RAM ,500GB HDD

### **7. Theory Concepts:**

#### **Advanced Macro Facilities:**

- (1) AIF
- (2) AGO
- (3) Sequential Symbol
- (4) Expansion time variable

### **(1) AIF**

Use the AIF instruction to branch according to the results of a condition test. You can thus alter the sequence in which source program statements or macro definition statements are processed by the assembler.

The AIF instruction also provides loop control for conditional assembly processing, which lets you control the sequence of statements to be generated.

It also lets you check for error conditions and thereby to branch to the appropriate MNOTE instruction to issue an error message.

### **(2) AGO**

The AGO instruction branches unconditionally. You can thus alter the sequence in which your assembler language statements are processed. This provides you with final exits from conditional assembly loops.

### **3) Sequence Symbols**

You can use a sequence symbol in the name field of a statement to branch to that statement during conditional assembly processing, thus altering the sequence in which the assembler processes your conditional assembly and macro instructions. You can select the model statements from which the assembler generates assembler language statements for processing at assembly time.

A sequence symbol consists of a period (.) followed by an alphabetic character, followed by 0 to 61 alphanumeric characters.

#### **Examples:**

.BRANCHING\_LABEL#1

.A

Sequence symbols can be specified in the name field of assembler language statements and model statements; however, sequence symbols must not be used as name entries in the following assembler instructions:

ALIAS	EQU	OPSYN	SETC
AREAD	ICTL	SETA	SETAF
CATTR	LOCTR	SETB	SETCF
DXD			

Also, sequence symbols cannot be used as name entries in macro prototype instructions, or in any instruction that already contains an ordinary or a variable symbol in the name field.

Sequence symbols can be specified in the operand field of an AIF or AGO instruction to branch to a statement with the same sequence symbol as a label

#### **4) Expansion Time Variables**

**Note: - write theory from book or notes.**

#### **- Data Structures of Two Pass Macros:**

1] Input Source Program for pass- II . It is produced by pass – I.

2] Macro Definition Table: (MDT) produced by pass - I

- Location Counter(where MDTP points to start position of macro)
- Opcode
- Rest (i.e it will contain the other part than opcodes used in macro).

3] Macro Name Table : (MNT) produced by pass - I

- macro number(i.e pointer referenced from MNTP)
- Name of macros
- MDTP (i.e points to start position to MDT)
- 

4] MNTP (macro name table pointer) gives number of entries in MNT.

5] Argument List Array:

- Index given to parameter
- Name of parameter

Which gives association between integer indices & actual parameters?

6] Source Program with macro-calls expanded. This is output of pass- II.

7] MDTP (macro definition table pointer) gives the address of macro definition in macro definition table.

#### **Algorithm:**

Take Input from Pass - I

Examine all statements in the assembly source program to detect macro calls. For Each Macro call:

- (a) Locate the macro name in MNT.
- (b) Establish correspondence between formal parameters & actual parameters.
- (c) Obtain information from MNT regarding position of the macro definition in MDT.
- (d) Expand the macro call by picking up model statements from MDT.

### Example for pass I of macro processor: -

#### Assembly Code Segment:-

MACRO

INCR &X, Y, &REG=AREG

MOVER &REG, &Y

ADD &REG, &Y

MOVEM &REG,

&X MEND

MACRO

DECR &A,&B,&REG=BREG

MOVER &REG,&A

ADD &REG,&B

MOVEM &REG,&A

MEND

START 200

READ N1

READ N2

INCR N1, 2,REG=CREG

DECR N1,N2

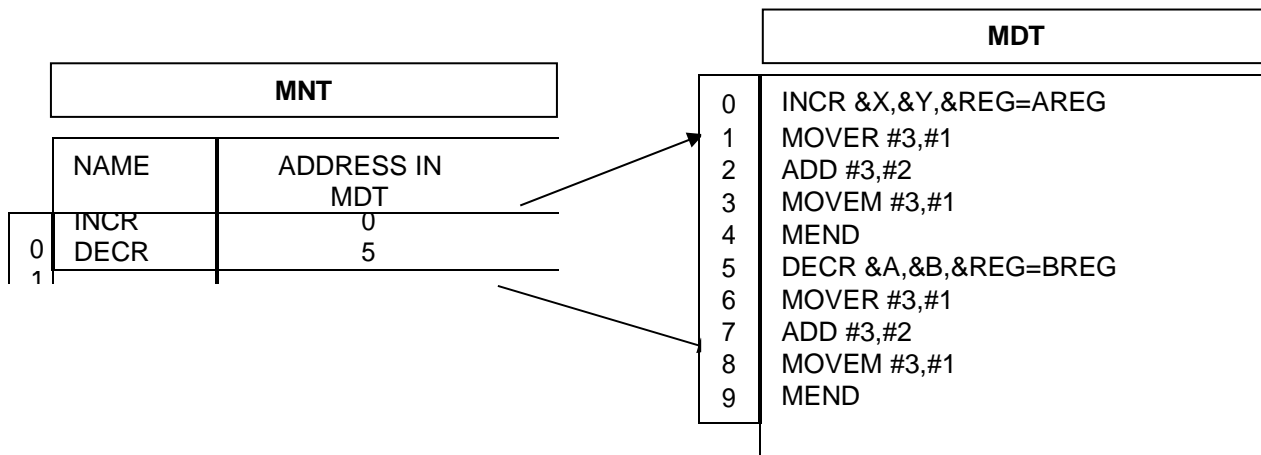
STOP

N1 DS 1

N2 DS 2

END

Output of Pass-I of Macro Processor:-



**#1 – First Parameter**

**#2 – Second Parameter**

**#3 – Third Parameter**

Pass – II of macro pre-processor will create the argument list array , every time there is call to macro , & expand macro.

(1) Macro Call:-

INCR &X, &Y, &REG=AREG

ALA:-

N1  
N2  
AREG  
G

Attach expansion code of this macro in output as following:-

MOVER AREG,N1

ADD AREG, N2

MOVEM AREG, N1

(2) Macro Call:-

INCR &A, &B, &REG=BREG

ALA:-

N1  
N2  
BREG  
G

Attach expansion code of this macro in output as following:-

MOVER BREG, N1

ADD BREG, N2

MOVEM BREG, N1

Expanded Source file at the end of pass – II:

START 200

READ N1

READ N2

MOVER AREG, N1

ADD AREG, N2

N2 MOVEM AREG, N1

Expansion of INCR N1,

MOVER BREG, N1

ADD BREG, N2

Expansion of DECR N1,

N2 MOVEM BREG, N1

STOP

N1 DS 1

N2 DS 2

END

## **8. Conclusion :**

Write your own conclusion.

### **Questions: [Write short answer]**

1. What are MDT and MNT?
2. What is Argument List Array?
3. Working of Pass-2 macro processor

## **GROUP – B**

### **ASSIGNMENT NO: 04**

#### **1. Title:**

Write a program to simulate CPU Scheduling Algorithms: FCFS, SJF (Preemptive), Priority (Non-Preemptive) and Round Robin (Preemptive).

#### **2. Objectives:**

- To understand OS & SCHEDULLING Concepts
- To implement Scheduling FCFS, SJF, RR & Priority algorithms
- To study about Scheduling and scheduler

#### **3. Problem Statement:**

Write a Java program (using OOP features) to implement following scheduling algorithms: FCFS, SJF, Priority and Round Robin.

#### **4. Outcomes:**

After completion of this assignment students will be able to:

- Knowledge Scheduling policies
- Compare different scheduling algorithms

#### **5. Software Requirements:**

JDK/Eclipse

#### **6. Hardware Requirement:**

- M/C Lenovo Think center M700 Ci3,6100,6th Gen. H81, 4GB RAM ,500GB HDD

#### **7. Theory Concepts:**

##### **CPU Scheduling:**

- CPU scheduling refers to a set of policies and mechanisms built into the operating systems that govern the order in which the work to be done by a computer system is completed.
- Scheduler is an OS module that selects the next job to be admitted into the system and next process to run.
- The primary objective of scheduling is to optimize system performance in accordance with the criteria deemed most important by the system designers.

##### **What is scheduling?**

Scheduling is defined as the process that governs the order in which the work is to be done. Scheduling is done in the areas where more no. of jobs or works are to be performed. Then it requires some plan i.e. scheduling that means how the jobs are to be performed i.e. order. CPU scheduling is best example of scheduling.

### What is scheduler?

1. Scheduler in an OS module that selects the next job to be admitted into the system and the next process to run.
2. Primary objective of the scheduler is to optimize system performance in accordance with the criteria deemed by the system designers. In short, scheduler is that module of OS which schedules the programs in an efficient manner.

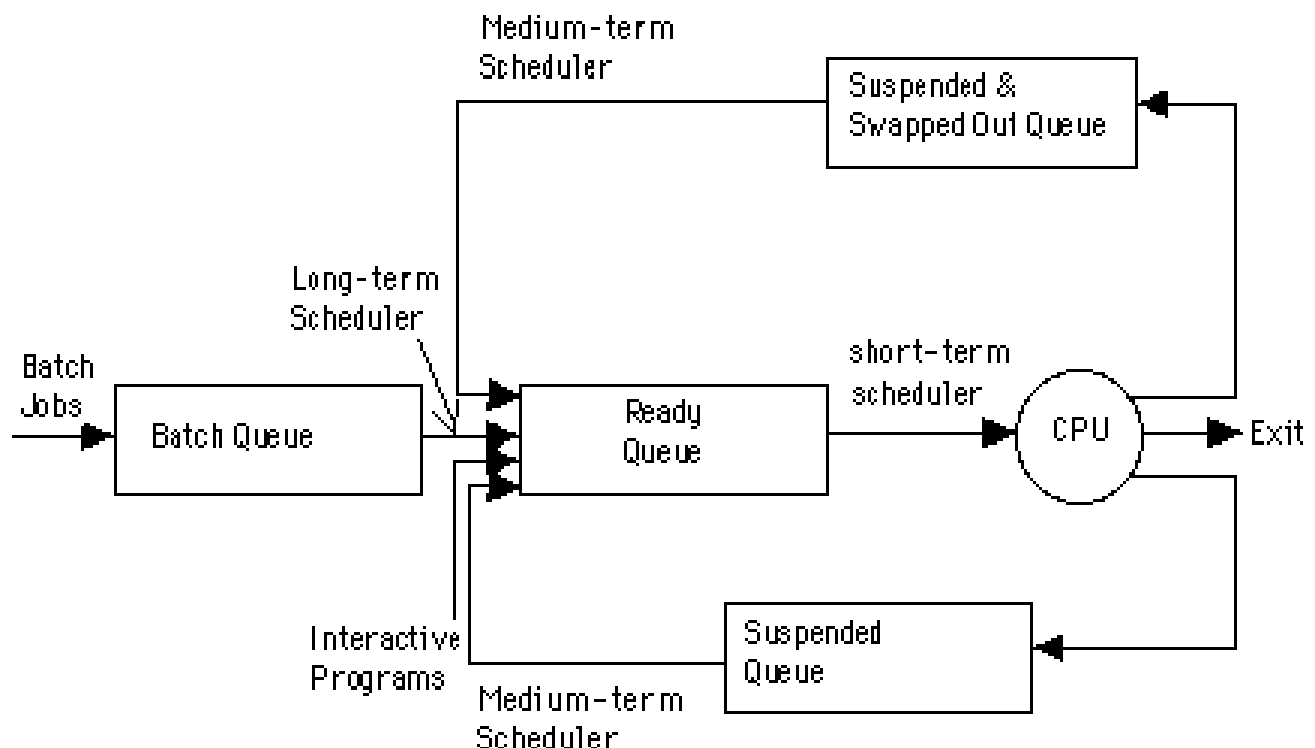
### Necessity of scheduling

- Scheduling is required when no. of jobs are to be performed by CPU.
- Scheduling provides mechanism to give order to each work to be done.
- Primary objective of scheduling is to optimize system performance.
- Scheduling provides the ease to CPU to execute the processes in efficient manner.

### Types of schedulers

In general, there are three different types of schedulers which may co-exist in a complex operating system.

- Long term scheduler
- Medium term scheduler
- Short term scheduler.





### **Long Term Scheduler**

- The long term scheduler, when present works with the batch queue and selects the next batch job to be executed.
- Batch is usually reserved for resource intensive (processor time, memory, special I/O devices) low priority programs that may be used fillers of low activity of interactive jobs.
- Batch jobs usually also contains programmer-assigned or system-assigned estimates of their resource needs such as memory size, expected execution time and device requirements.
- Primary goal of long term scheduler is to provide a balanced mix of jobs.

### **Medium Term Scheduler**

- After executing for a while, a running process may be suspended by making an I/O request or by issuing a system call.
- When number of processes becomes suspended, the remaining supply of ready processes in systems where all suspended processes remains resident in memory may become reduced to a level that impairs functioning of schedulers.
- The medium term scheduler is in charge of handling the swapped out processes.
- It has little to do while a process is remained as suspended.

### **Short Term Scheduler**

- The short term scheduler allocates the processor among the pool of ready processes resident in the memory.
- Its main objective is to maximize system performance in accordance with the chosen set of criteria.
- Some of the events introduced thus far that cause rescheduling by virtue of their ability to change the global system state are:
  - Clock ticks
  - Interrupt and I/O completions
  - Most operational OS calls
  - Sending and receiving of signals
  - Activation of interactive programs.
- Whenever one of these events occurs, the OS involves the short term scheduler.

### **Scheduling Criteria :**

#### **☐ CPU Utilization:**

Keep the CPU as busy as possible. It ranges from 0 to 100%. In practice, it ranges from 40 to 90%.

#### **☐ Throughput:**

Throughput is the rate at which processes are completed per unit of time.

## ▬ **Turnaround time:**

This is the how long a process takes to execute a process. It is calculated as the time gap between the submission of a process and its completion.

## ▬ **Waiting time:**

Waiting time is the sum of the time periods spent in waiting in the ready queue.

## ▬ **Response time:**

Response time is the time it takes to start responding from submission time. It is calculated as the amount of time it takes from when a request was submitted until the first response is produced.

## **Non-preemptive Scheduling :**

In non-preemptive mode, once if a process enters into running state, it continues to execute until it terminates or blocks itself to wait for Input/Output or by requesting some operating system service.

## **Preemptive Scheduling :**

In preemptive mode, currently running process may be interrupted and moved to the ready State by the operating system.

When a new process arrives or when an interrupt occurs, preemptive policies may incur greater overhead than non-preemptive version but preemptive version may provide better service.

It is desirable to maximize CPU utilization and throughput, and to minimize turnaround time, waiting time and response time.

## **Types of scheduling Algorithms**

- In general, scheduling disciplines may be pre-emptive or non-pre-emptive .
- In batch, non-pre-emptive implies that once scheduled, a selected job turns to completion.

There are different types of scheduling algorithms such as:

- FCFS(First Come First Serve)
- SJF(Short Job First)
- Priority scheduling
- Round Robin Scheduling algorithm

## **First Come First Serve Algorithm**

- FCFS is working on the simplest scheduling discipline.
- The workload is simply processed in an order of their arrival, with no pre-emption.
- FCFS scheduling may result into poor performance.
- Since there is no discrimination on the basis of required services, short jobs may considerable in turn around delay and waiting time.

## Advantages

- Better for long processes
- Simple method (i.e., minimum overhead on processor)
- No starvation

## Disadvantages

- Convoy effect occurs. Even very small process should wait for its turn to come to utilize the CPU. Short process behind long process results in lower CPU utilization.
- Throughput is not emphasized.

## First Come, First Served

<u>Process</u>	<u>Burst Time</u>
<i>P1</i>	24
<i>P2</i>	3
<i>P3</i>	3

- Suppose that the processes arrive in the order: *P1*, *P2*, *P3*
- The Gantt Chart for the schedule is:



- Waiting time for *P1* = 0; *P2* = 24; *P3* = 27
- Average waiting time:  $(0 + 24 + 27)/3 = 17$

**Note :** solve complete e.g. as we studied in practical(above is just sample e.g.). you can take any e.g.

## Shortest Job First Algorithm :

- This is also known as **shortest job first**, or SJF
- This is a non-preemptive, pre-emptive scheduling algorithm.
- Best approach to minimize waiting time.
- Easy to implement in Batch systems where required CPU time is known in advance.
- Impossible to implement in interactive systems where required CPU time is not known.
- The processor should know in advance how much time process will take.

## Advantages

- It gives superior turnaround time performance to shortest process next because a short job is given immediate preference to a running longer job.
- Throughput is high.

## Disadvantages

- Elapsed time (i.e., execution-completed-time) must be recorded, it results an additional overhead on the processor.
- Starvation may be possible for the longer processes.

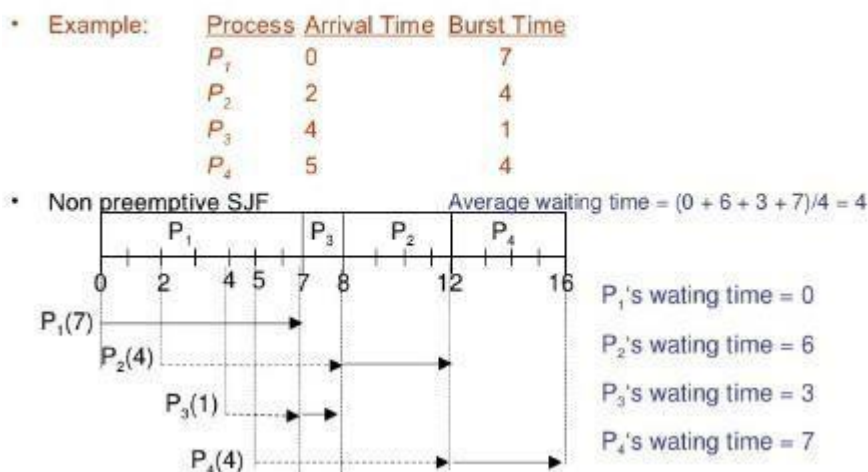
**This algorithm is divided into two types:**

- Pre-emptive SJF
- Non-pre-emptive SJF

### • Pre-emptive SJF Algorithm:

In this type of SJF, the shortest job is executed 1st. the job having least arrival time is taken first for execution. It is executed till the next job arrival is reached.

## Shortest Job First Scheduling



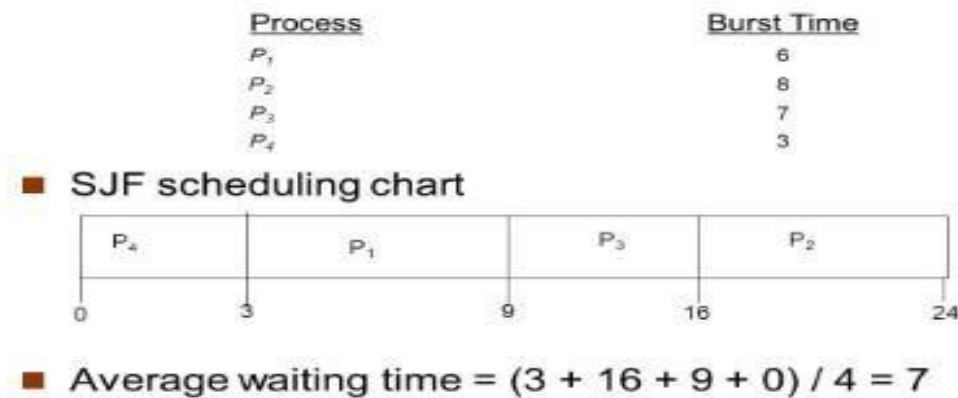
**Note :** solve complete e.g. as we studied in practical(above is just sample e.g.). you can take any e.g.

### Non-pre-emptive SJF Algorithm:

In this algorithm, job having less burst time is selected 1st for execution. It is executed for its total burst time and then the next job having least burst time is selected.



## Example of SJF



**Note :** solve complete e.g. as we studied in practical(above is just sample e.g.). you can take any e.g.

### Round Robin Scheduling :

- Round Robin is the preemptive process scheduling algorithm.
- Each process is provided a fix time to execute, it is called a **quantum**.
- Once a process is executed for a given time period, it is preempted and other process executes for a given time period.
- Context switching is used to save states of preempted processe

#### Advantages

- Round-robin is effective in a general-purpose, times-sharing system or transaction-processing system.
- Fair treatment for all the processes.
- Overhead on processor is low.
- Overhead on processor is low.
- Good response time for short processes.

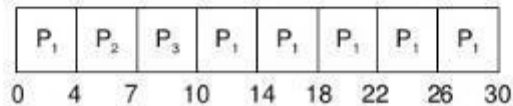
#### Disadvantages

- Care must be taken in choosing quantum value.
- Processing overhead is there in handling clock interrupt.
- Throughput is low if time quantum is too small.

# Round Robin

<u>Process</u>	<u>Burst Time</u>
<i>P1</i>	24
<i>P2</i>	3
<i>P3</i>	3

- Quantum time = 4 milliseconds
- The Gantt chart is:



- Average waiting time =  $\{[0+(10-4)]+4+7\}/3 = 5.6$

**Note :** solve complete e.g. as we studied in practical(above is just sample e.g.). you can take any e.g.

## Priority Scheduling :

- Priority scheduling is a non-preemptive algorithm and one of the most common scheduling algorithms in batch systems.
- Each process is assigned a priority. Process with highest priority is to be executed first and so on.
- Processes with same priority are executed on first come first served basis.
- Priority can be decided based on memory requirements, time requirements or any other resource requirement.

### Advantage

- Good response for the highest priority processes.

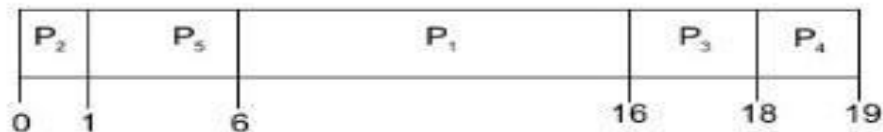
### Disadvantage

- Starvation may be possible for the lowest priority processes.

# Priority

<u>Process</u>	<u>Burst Time</u>	<u>Priority</u>
<i>P1</i>	10	3
<i>P2</i>	1	1
<i>P3</i>	2	4
<i>P4</i>	1	5
<i>P5</i>	5	2

## ▪ Gantt Chart



▪ Average waiting time =  $(6 + 0 + 16 + 18 + 1) / 5 = 8.2$

Note : solve complete e.g. as we studied in practical(above is just sample e.g.). you can take any e.g.

## 8. Algorithms(procedure) :

### FCFS :

Step 1: Start the process

Step 2: Accept the number of processes in the ready Queue

Step 3: For each process in the ready Q, assign the process id and accept the CPU burst time

Step 4: Set the waiting of the first process as '0' and its burst time as its turn around time

Step 5: for each process in the Ready Q calculate

(a) Waiting time for process(n)= waiting time of process (n-1) + Burst time of process(n-1)

(b) Turn around time for Process(n)= waiting time of Process(n)+ Burst time for process(n)

Step 6: Calculate

(a) Average waiting time = Total waiting Time / Number of process

(b) Average Turnaround time = Total Turnaround Time / Number of process

Step 7: Stop the process

### SJF :

Step 1: Start the process

Step 2: Accept the number of processes in the ready Queue

Step 3: For each process in the ready Q, assign the process id and accept the CPU burst time

Step 4: Start the Ready Q according the shortest Burst time by sorting according to lowest to highest burst time.

Step 5: Set the waiting time of the first process as '0' and its turnaround time as its burst time.

Step 6: For each process in the ready queue, calculate

(c) Waiting time for process(n)= waiting time of process (n-1) + Burst time of process(n-1)

(d) Turn around time for Process(n)= waiting time of Process(n)+ Burst time for process(n)

Step 6: Calculate

(c) Average waiting time = Total waiting Time / Number of process

(d) Average Turnaround time = Total Turnaround Time / Number of process

Step 7: Stop the process

### **RR :**

Step 1: Start the process

Step 2: Accept the number of processes in the ready Queue and time quantum (or) time slice

Step 3: For each process in the ready Q, assign the process id and accept the CPU burst time

Step 4: Calculate the no. of time slices for each process where

No. of time slice for process(n) = burst time process(n)/time slice

Step 5: If the burst time is less than the time slice then the no. of time slices =1.

Step 6: Consider the ready queue is a circular Q, calculate

(a) Waiting time for process(n) = waiting time of process(n-1)+ burst time of process(n-1 ) + the time difference in getting the CPU from process(n-1)

(b) Turn around time for process(n) = waiting time of process(n) + burst time of process(n)+ the time difference in getting CPU from process(n).

Step 7: Calculate

(e) Average waiting time = Total waiting Time / Number of process

(f) Average Turnaround time = Total Turnaround Time / Number of process

Step 8: Stop the process.

### **Priority Scheduling :**

#### **Algorithms :**

Step 1: Start the process

Step 2: Accept the number of processes in the ready Queue

Step 3: For each process in the ready Q, assign the process id and accept the CPU burst time, priority

Step 4: Start the Ready Q according the priority by sorting according to lowest to highest burst time and process.

Step 5: Set the waiting time of the first process as '0' and its turnaround time as its burst time.



Step 6: For each process in the ready queue, calculate

- (e) Waiting time for process(n)= waiting time of process (n-1) + Burst time of process(n-1)
- (f) Turn around time for Process(n)= waiting time of Process(n)+ Burst time for process(n)

Step 6: Calculate

- (g) Average waiting time = Total waiting Time / Number of process
- (h) Average Turnaround time = Total Turnaround Time /

Number of process

Step 7: Stop the process

## 9. Conclusion:

Hence we have studied that-

- CPU scheduling concepts like context switching, types of schedulers, different timingparameter like waiting time, turnaround time, burst time, etc.
- Different CPU scheduling algorithms like FIFO, SJF,Etc.
- FIFO is the simplest for implementation but produces large waiting times and reducessystem performance.
- SJF allows the process having shortest burst time to execute first.

## References :

<https://www.studytonight.com/operating-svstem/cpu-scheduling>  
<https://www.go4expert.com/articles/types-of-scheduling-t22307/>  
[https://en.wikipedia.org/wiki/Scheduling\\_\(computing\)](https://en.wikipedia.org/wiki/Scheduling_(computing))  
[https://www.tutorialspoint.com/operating\\_system/os\\_process\\_scheduling\\_algorithms.htm](https://www.tutorialspoint.com/operating_system/os_process_scheduling_algorithms.htm)

## Questions: [Write short answer ]

1. Scheduling? List types of scheduler & scheduling.
2. List and define scheduling criteria.
3. Define preemption & non-preemption.
4. State FCFS, SJF, Priority & Round Robin scheduling.
5. Compare FCFS, SJF, RR, Priority

## ASSIGNMENT NO: 05

### 1. Title:

Write a Program to simulate Page replacement algorithm.

1. Least Recently Used (LRU)
2. Optimal algorithm

### 2. Objectives:

- To understand concept of paging.
- To Learn different page replacement algorithms.

### 3. Problem Statement:

Write a program to implement paging simulation.

### 4. Outcomes:

After completion of this assignment students will be able to:

- Understand the Programming language of Java
- Understand the concept of Paging

### 5. Software Requirements:

- Linux OS, JDK1.7

### 6. Hardware Requirement:

- 4GB RAM ,500GB HDD

### 7.Theory Concepts:

#### Paging

Paging is a memory-management scheme that permits the physical-address space of a process to be noncontiguous. [Page Replacement Algorithms](#) In operating systems that use paging for memory management, page replacement algorithm are needed to decide which page needed to be replaced when new page comes in. Whenever a new page is referred and not present in memory, page fault occurs and Operating System replaces one of the existing pages with newly needed page. Different page replacement algorithms suggest different ways to decide which page to replace. The target for all algorithms is to reduce number of page faults.

In paging, the physical memory is divided into fixed-sized blocks called **page frames** and logical memory is also divided into fixed-size blocks called **pages** which are of same size as that of page frames.

When a process is to be executed, its pages can be loaded into any unallocated frames (not necessarily contiguous) from the disk.

Consider the size of logical address space is  $2^m$ . Now, if we choose a page size of  $2^n$ , then  $n$  bits will specify the page offset and  $m-n$  bits will specify the page number.

Consider a system that generates logical address of 16 bits and page size is 4 KB. How many bits would specify the page number and page offset?

### How a logical address is translated into a physical address:

In paging, address translation is performed using a mapping table, called **Page Table**. The operating system maintains a page table for each process to keep track of which page frame is allocated to which page. It stores the frame number allocated to each page and the page number is used as index to the page table.

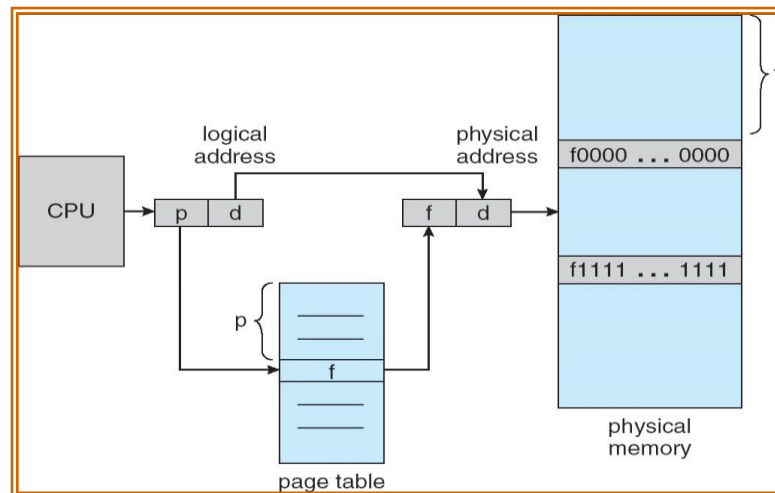


Fig1 paging

### LRU

- Replaces the page that has not been referenced for the longest time:
  - By the principle of locality, this should be the page least likely to be referenced in the near future.
  - Performs nearly as well as the optimal policy.

reference string

7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

7	7	7	2		2		4	4	4	0			1		1		1		
	0	0	0		0		0	0	3	3			3		0		0		
		1	1		3		3	2	2	2			2		2		7		

page frames

Example of LRU

## Optimal

Optimal Page Replacement refers to the removal of the page that will not be used in the future, for the longest period of time.

reference string

7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

7	7	7	2		2				2				2				7		
	0	0	0		0		4		0				0				0		
		1	1		3		3		3				1				1		

page frames

## First In First Out (FIFO) page replacement algorithm –

This is the simplest page replacement algorithm. In this algorithm, operating system keeps track of all pages in the memory in a queue, oldest page is in the front of the queue. When a page needs to be replaced page in the front of the queue is selected for removal.

**Example -1.** Consider page reference string 1, 3, 0, 3, 5, 6 and 3 page slots. Initially all slots are empty, so when 1, 3, 0 came they are allocated to the empty slots → 3 **Page Faults**. when 3 comes, it is already in memory so → 0 **Page Faults**. Then 5 comes, it is not available in memory so it replaces the oldest page slot i.e 1. → 1 **Page Fault**. Finally 6 comes, it is also not available in memory so it replaces the oldest page slot i.e 3 → 1 **Page Fault**.

So total page faults = 5.

**Example -2.** Consider the following reference string: 0, 2, 1, 6, 4, 0, 1, 0, 3, 1, 2, 1. Using FIFO page replacement algorithm –

0	2	1	6	4	0	1	0	3	1	2	1
0	0	0	0	4	4			4	4	2	
	2	2	2	2	0		hit	0	0	0	
		1	1	1	1	hit		3	3	3	
			6	6	6			6	1	1	hit

So, total number of page faults = 9. Given memory capacity (as number of pages it can hold) and a string representing pages to be referred, write a function to find number of page faults.

**Implementation** – Let capacity be the number of pages that memory can hold. Let set be the current set of pages in memory.

1- Start traversing the pages.

i) If set holds less pages than capacity.

a) Insert page into the set one by one until  
the size of set reaches capacity or all  
page requests are processed.

b) Simultaneously maintain the pages in the  
queue to perform FIFO.

c) Increment page fault

ii) Else

If current page is present in set, do nothing.

Else

a) Remove the first page from the queue  
as it was the first to be entered in  
the memory

b) Replace the first page in the queue with  
the current page in the string.

c) Store current page in the queue.

d) Increment page faults.

2. Return page faults.

## **9. Conclusion:**

The various memory management page replacement algorithms were studied and successfully implemented.

## **References:**

Andrew S. Tanenbaum, “Modern Operating Systems”, Second Edition, PHI.

[Chapter 3 topic 4.3.1, 4.4, 4.4.3, 4.4.6]

## **Questions: [Write short answer]**

1. What is page fault?

2. What is the difference between physical memory and logical memory?
3. Explain virtual memory.
4. What is the difference between paging and segmentation?

## GROUP C

### ASSIGNMENT NO: 06

#### Title: Create Project Plan

- Specify project name and start (or finish) date.
- Identify and define project tasks.
- Define duration for each project task.
- Define milestones in the plan
- Define dependency between tasks
- Define project calendar.
- Define project resources and specify resource type
- Assign resources against each task and baseline the project plan

#### Theory:

##### What is project planning?

Project planning is the process of defining the [project scope](#), objectives, and steps needed to get the work done. It's one of the most important processes in project management. The output of the project planning process is a project management plan.

##### What is a project management plan?

A project management plan also known as a project plan is a document that outlines the process your team will use to manage the project according to scope to meet its stated objectives. The purpose of a project plan is to map out the steps and resources it will take to complete a project on time and budget.

A project plan communicates vital information—such as deadlines, assignments, and [key milestones](#)—to all project stakeholders and is integral to project success. It is most commonly represented in the form of a [gantt chart](#) to make it easy to ensure work stays on track.

##### Project planning steps: How to write a good project plan

Poor planning can lead to some pretty ugly consequences from missed deadlines and budget overages to team burnout and client frustration. That's why it's important to establish a solid process you can use to plan any project.

Planning a project doesn't have to be difficult. These basic project planning steps can help you write a plan that's both realistic and on target.

1. [Start with research and preplanning](#)
2. [Draft a rough outline of your project plan](#)
3. [Build out your detailed project schedule](#)
4. [Present and confirm your plan](#)

## 5. [Execute your plan and adjust as needed](#)

### Step 1: Start with research and preplanning

A project plan is more than a dry document with dates. It's the story of your project, and you don't want it to be a tall tale! So make sure you know all the facts before you start creating a project plan.

#### Understand the project scope and value

Understanding the ins and outs of the project will help you determine the best process and identify any snags that might get in the way of success. Conduct your own research to dig deeper on:

- Project goals and outcomes
- Partnerships and outlying dependencies
- Potential issues and risks

Dive into any communications that are relevant to the project. Review the [scope of work](#) and related documents (maybe an RFP or notes from sales calls or meetings with your client team). Be thorough in your research to uncover critical project details, and ask thoughtful questions before you commit to anything.

#### Interview key stakeholders

If you want to dazzle stakeholders with a stellar project delivery, you've got to know how they work and what they expect. Schedule time with your main project contact, and ask them some tough questions about process, organizational politics, and general risks before creating a project plan.

This will give project stakeholders confidence that your team has the experience to handle any difficult personality or situation. It also shows you care about the success of the project from the start.

Be sure to discuss these things with your stakeholders:

- Product ownership and the decision-making process
- Stakeholder interest/involvement levels
- Key outages, meetings, deadlines, and driving factors
- Related or similar projects, goals, and outcomes
- The best way to communicate with partners and stakeholders

#### Get to know your team

The last step in the research phase is to take time to learn more about the people who'll be responsible for the work. Sit down with your team and get to know their:

- Expertise
- Interests
- Collaboration and communication styles



- Availability and workload

Understanding these basics about your team will help you craft a thoughtful plan that takes their work styles and bandwidth into consideration. After all, a happy team delivers better projects.

## Step 2: Draft a rough outline of your project plan

Now that you've gathered the basic project details, the next step is to knock out a rough draft of your plan. Take some time to think about the discussions you had in the pre-planning phase and the approach your team might take to meet the project goals.

### Sketch out the main components of your project plan

Sit down with a pen and paper (or a whiteboard), and outline how the project should work at a high level. Be sure you have a calendar close by to check dates.

If you're at a loss for where to begin, start with the who, what, when, and how of the project. Any solid project plan should answer these questions:

- What are the major deliverables?
- How will we get to those deliverables and the deadline?
- Who's on the project team, and what role will they play in those deliverables?
- When will the team meet milestones?
- When will other members of the team play a role in contributing to or providing feedback on those deliverables?

A first outline can be very rough and might look something like a [work breakdown structure](#), as noted in [our chapter on project estimation](#). Make sure your project plan outline includes the following components:

- Deliverables and the tasks taken to create them
- Your client's approval process
- Timeframes associated with tasks/deliverables
- Ideas on resources needed for tasks/deliverables
- A list of the assumptions you're making in the plan
- A list of absolutes as they relate to the project budget and/or deadlines

Considering these elements will help you avoid surprises or at least minimize them. And remember, you're doing this as a draft so you can use it as a conversation starter for your team. It's not final yet!

### Get input from your team on process, effort, and timing

You don't want to put yourself or your team in an awkward position by not coming to a consensus on the approach before presenting it to your client. That's why a project manager can't be the only one writing a project plan.

Once you've sketched out a basic outline of your plan, take those rough ideas and considerations to your team. This enables you to invite discussion about what might work rather than simply dictating a process. After all, every project must begin with clear communication of the project goals and the effort required to meet them.

Be sure to get input from your team on how they can complete the tasks at hand without killing the budget and the team's morale. As a project manager, you can decide on [Agile vs. Waterfall approaches](#), but when it comes down to it, you need to know that the team can realistically execute the plan.

You can also use this project plan review time to question your own thinking and push the team to take a new approach to the work. For example, if you're working on a [website design project plan](#), could designers start creating visual concepts while the wireframes are being developed? Or can you have two resources working on the same task at once?

Running ideas by the team and having an open dialogue about the approach not only helps you build a more accurate project plan. It gets everyone thinking about the project in the same terms. This type of buy-in and [communication builds trust](#) and gets people excited about working together to solve a goal. It can work wonders for the greater good of your team and project.

### **Step 3: Build out your detailed project schedule**

You should feel comfortable enough at this point to put together a rock-solid project schedule using whatever tool works for you. (Ahem, [TeamGantt works nicely for a lot of happy customers](#).)

#### **Build your project plan**

Any good [online project planning tool](#) will help you formalize your thoughts and lay them out in a consistent, visual format that's easy to follow and track. Make sure tasks, durations, milestones, and dates are crystal-clear, and try to keep your project plan simple. The easier it is to read, the better!

Be as flexible as possible when it comes to how your project plan is presented. There's no absolute when it comes to how to format your project plan as long as you and your team understand what goes into one.

Remember, people absorb information differently. While you might be partial to a [gantt chart](#), others might prefer to view tasks in a list, calendar, or even a [kanban board](#). You can make all of those variations work if you've taken the steps to create a solid plan.

TeamGantt gives you the ability to quickly and easily build and adjust a project plan using a [simple drag and drop feature](#). Plus, it comes with customizable views to fit every team member's work style.

### **Step 4: Present and confirm your plan**

You're almost finished! You've done your research, outlined your approach, discussed it with your team, and built your formal project plan.

Now it's time to do your due diligence. It's easy to throw stuff in a plan, but you have to make sure you get it right.

#### ***Run your final plan by your internal team***

Your team needs to know the reality of your plan as it stands after you've built it out in TeamGantt. And you want to be sure they're comfortable committing to the details. If they don't, things will quickly fall apart!

Always review your final plan with your team before delivering it to stakeholders. Why? Because things like dates and tasks—and even assignments—will shift as you formalize the rough sketch of your plan.

Here are a few things you'll want to discuss with your team as you review the final plan together:

- Review times

- Team work times
- Dependencies
- Time off, meetings, and milestones
- The final deadline
- Any assumptions you've made
- Major changes since your last talk

There's nothing more embarrassing than delivering a plan with an error or a promise you can't keep. Taking a few minutes to get buy-in from your team will give everyone peace of mind about your plan.

### ***Review your project plan with stakeholders***

Once you've confirmed the plan with your team and have their full sign-off, you're ready to [share your project plan with stakeholders](#).

When delivering your project plan, make sure you provide an executive summary. This might come in the form of a [project brief](#) or [project charter](#). A short recap of the overall methodology, resources, assumptions, deadlines, and related review times will help you convey what the plan means to the project and everyone involved.

Project plans can be daunting, so schedule time to [present your project plan](#) to your stakeholders at a high level. Here are some things you'll want to point out about your plan during this review:

- Overall process and pacing
- Major deliverables and timing
- The time they'll have to review deliverables
- Overall timing for task groups or phases
- How far off you are from the deadline
- Wiggle room on the final deadline

If a stakeholder is interested in the day-to-day details, feel free to walk them through the plan line by line. Otherwise, start by explaining overall sections or phases, and be sure to come back to your plan at intervals throughout the project to remind them of tasks, next steps, and overall progress.

### **Step 5: Execute your plan and adjust as needed**

Some projects are smooth and easy to manage, and others are a complete nightmare that wake you up at 3 a.m. every other night. Thankfully, having a solid project plan is your best defense against project chaos once work gets underway.

### ***Work your plan***

Keep in mind that project plans are living documents. Projects change constantly, and someone has to stay on top of—and document—that change. Remember to:

- Update your plan regularly as work progresses and things change
- Communicate changes to your team, partners, and stakeholders
- Monitor and communicate risks as your project evolves

## How to create a project plan in TeamGantt

### 1. Enter your basic project details.

To create a new project plan in TeamGantt, click the *New Project* button in the upper right corner of the *My Projects* screen. Then enter your project name and start date, and select the days of the week you want to include in your plan. Click *Create a New Project*

Project name  
Spring Campaign

Start Date  
Today

Template  
Blank Project

Days in Week

Sun

Mon

Tue

Wed

Thu

Fri

Sat

Preview templates

Create new project

Import a project

If you have a project that you have worked on previously, you can import them below.

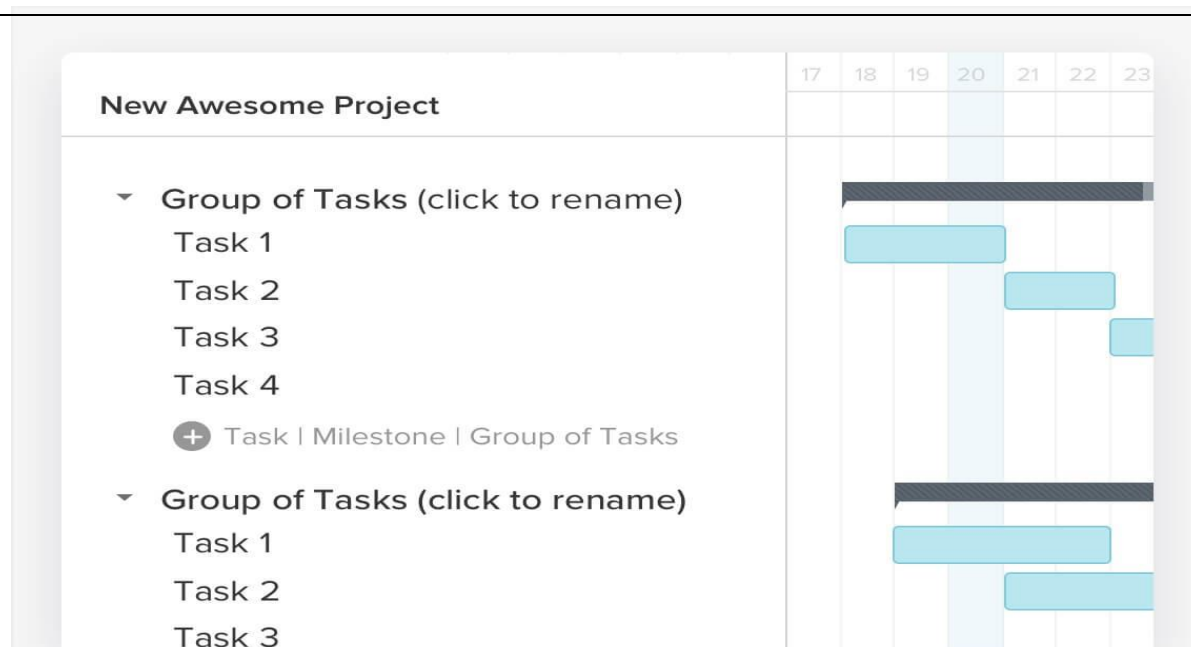
Upload CSV

Duplicate a project

Click *Create New Project* to move on to the next step.

### 2. List out your project tasks and milestones.

Now the real planning fun begins! Enter all the different tasks it will take to get the job done. If there are any key meetings, deliverable deadlines, or approvals, add those as milestones in your project plan.

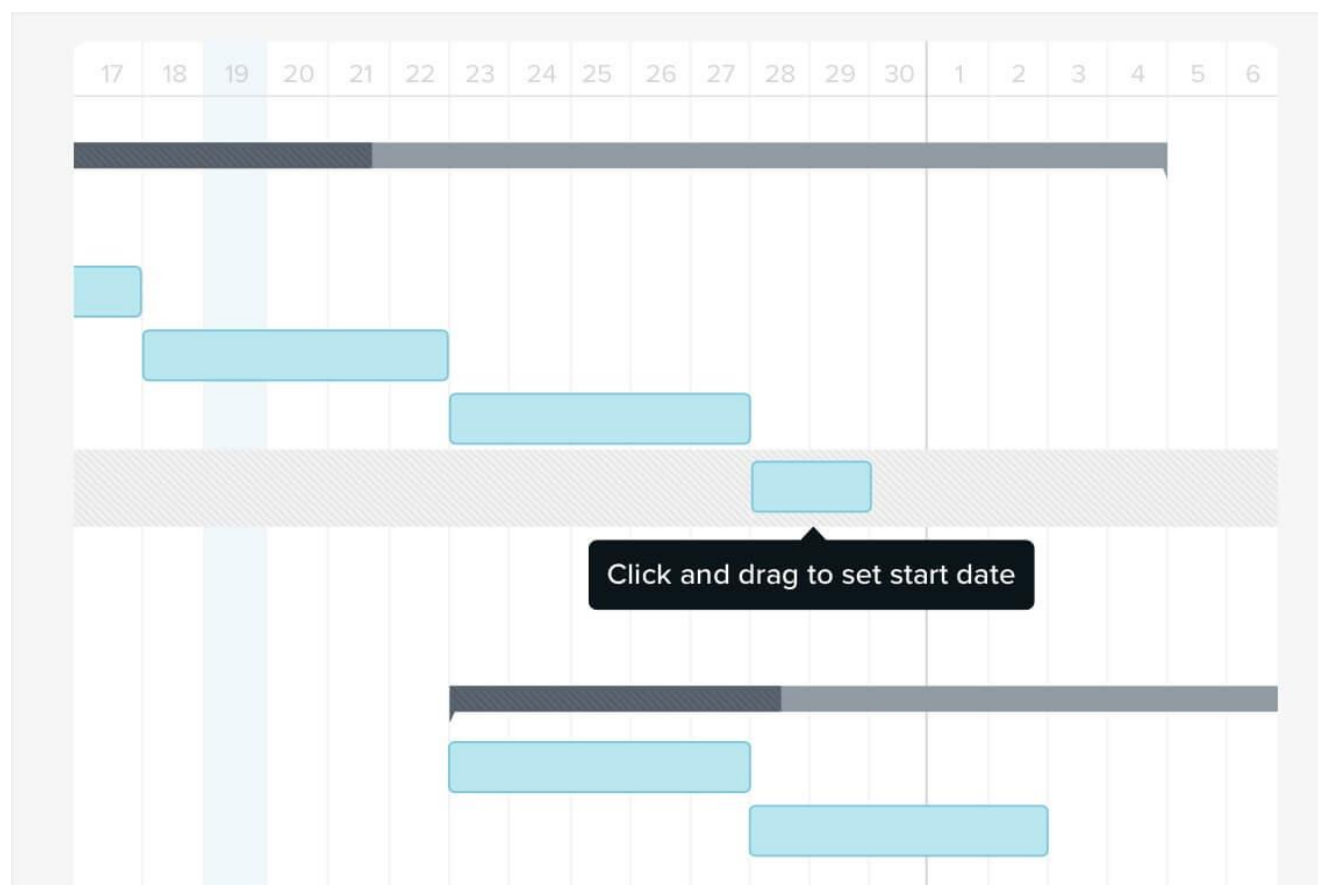


### 3. Organize tasks into subgroups.

Scrolling through one long list of tasks can be mind-numbing, even to the best of us. Break tasks down into phases or sections to ensure your project plan is easy to read and understand.

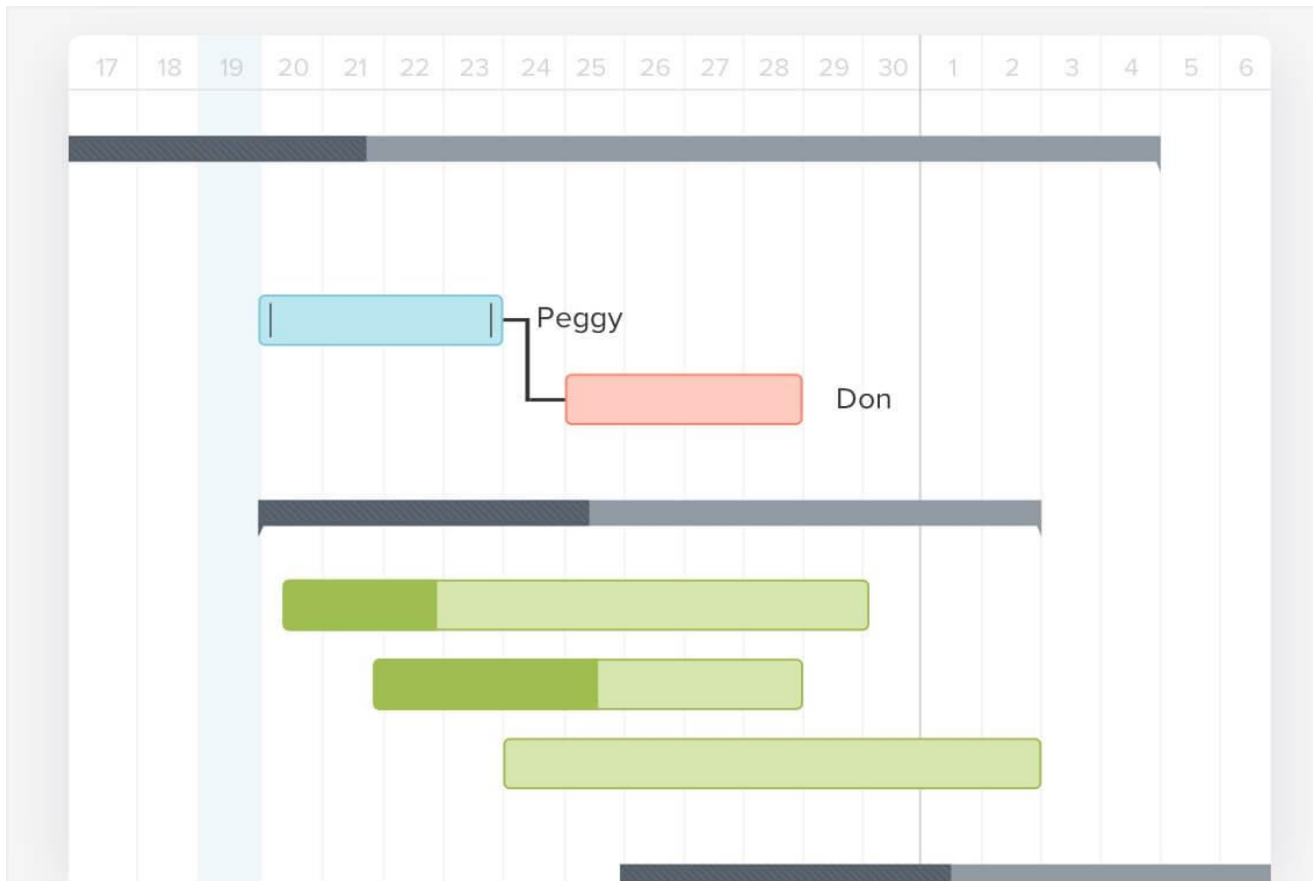
### 4. Add task durations and milestone dates to the project timeline.

A visual project plan makes it easy to see exactly what needs to get done by when. Make sure every task has a start and end date so nothing falls through the cracks. TeamGantt's drag and drop feature makes this planning step quick and easy.



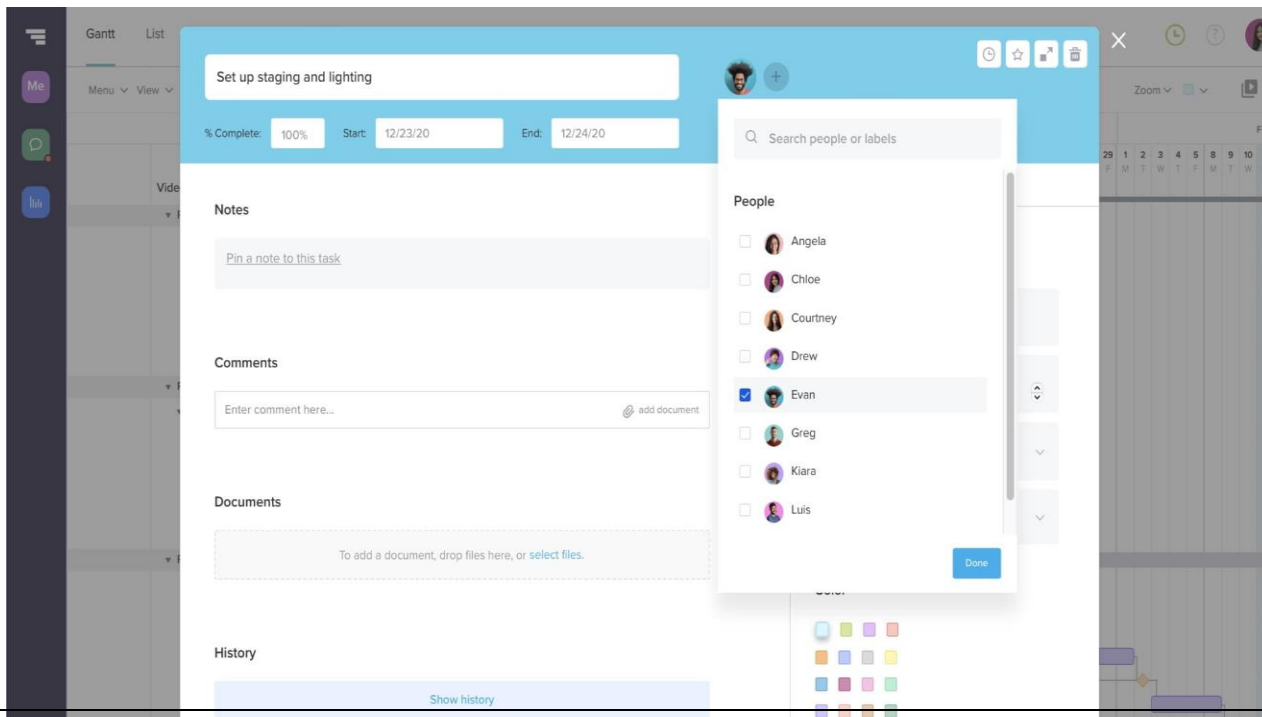
## 5. Connect related tasks with dependencies.

Adding dependencies between tasks ensures work gets done in the right order and also helps you plan for delay risks. If your plan shifts and you need to move tasks around, dependencies speed up the process.



## 6. Assign responsible team members to tasks.

That way there's no confusion about who's doing what, and [your team can update and manage their daily tasks](#). Don't forget to check team availability along the way to avoid overloading anyone with too much work.



## 7. Use the RACI chart to define task roles more clearly.

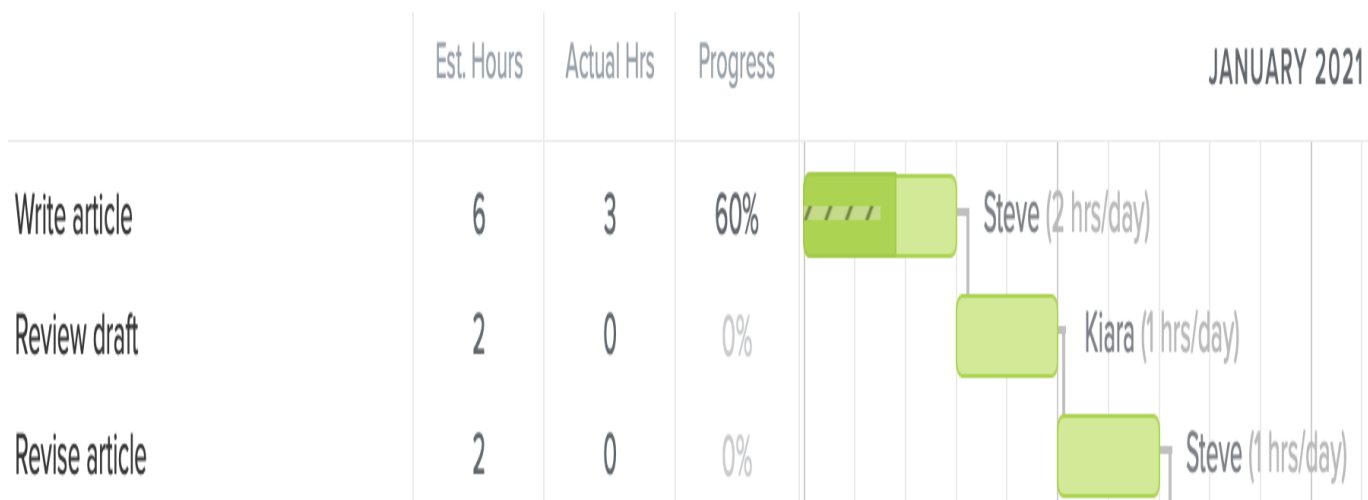
This feature takes accountability one step further by letting you assign more specific roles to each task: *Responsible, Accountable, Consulted, and Informed*.

The screenshot shows a project management interface with a sidebar on the left containing icons for 'Me', a chat bubble, and a calendar. The top navigation bar includes 'Gantt', 'List', 'Calendar', 'Board', 'Discussions', 'People', 'RACI' (selected), and 'More'. Below the navigation bar, there are filters for 'Menu', 'View', 'Filter', 'Search', and 'Hide Completed'. The main content area is titled 'TG Digital Marketing Campaign' and displays a RACI chart. The chart has columns for team members: Drew, Dwight, Henry, Holly, Jim, Kelsey, and Lindsey. The tasks are organized into three groups: Planning (0%), Content (0%), and Design (0%).

Task Group	Task	Drew	Dwight	Henry	Holly	Jim	Kelsey	Lindsey
Planning (click to edit) (0%)	Write campaign brief (click to edit)	I	R	A	I	C		I
	Team kickoff	I	R	A	I	I	I	I
	Creative brainstorm	C	R	C	C	C	C	C
Content (0%)	Write content	R	A	C				
	Review content	A	R	C	I	I	I	I
Design (0%)								

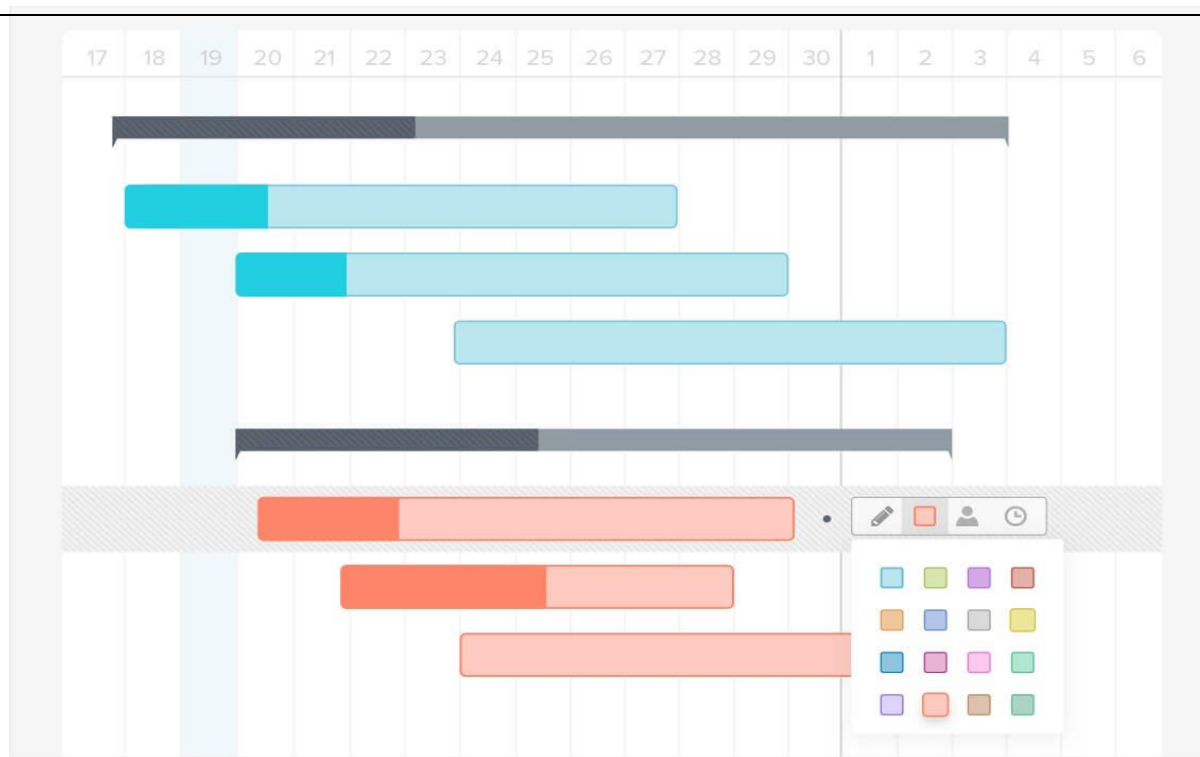
## 8. Add hourly estimates and/or points to each task.

This makes it easy to see the lift each task involves at a glance. Including hourly estimates in your project plan also enables you to [manage workloads](#) and [track overages](#) more accurately.



## 9. Color-code tasks for better scannability.

You can use colors to categorize tasks by project phase, priority, department, or team member—whatever makes visual sense to you and your team.



## 10. Add notes to clarify tasks or spell out important details.

There's no such thing as too much information if it means your team has what they need to deliver quality work on time. Use the *Notes* section of your *Discussion* tab to enter any pertinent details your team will find helpful.

The screenshot shows a project management interface. At the top, there's a header with 'Write article' and a link 'Request a progress update'. Below the header, there are tabs for 'All', 'Comments', and 'Documents'. The main content area shows a discussion thread. A post from Kiara, dated 12/14/20 @ 12:49 pm, says: '@Steve: Here's the creative brief for this article. Just let me know if you have any questions or need more info. Thanks!'. Below the text is a document attachment titled 'Creative Brief.docx' with a description 'Click to add a description'. At the bottom, there's a text input field with '@Steve' and a button 'add document'.

## 11. Upload important documents to the project.

This ensures project files are accessible to everyone in a centralized hub. You might attach your [scope document](#), [project requirements](#), [risk assessment matrix](#), or even a [creative brief](#) to guide your team to successful completion.



# How to make Gantt chart in Excel

Regrettably, Microsoft Excel does not have a built-in Gantt chart template as an option. However, you can quickly create a Gantt chart in Excel by using the bar graph functionality and a bit of formatting.

Please follow the below steps closely and you will make a simple Gantt chart in under 3 minutes. We will be using Excel 2010 for this Gantt chart example, but you can simulate Gantt diagrams in any Excel version in the same way.

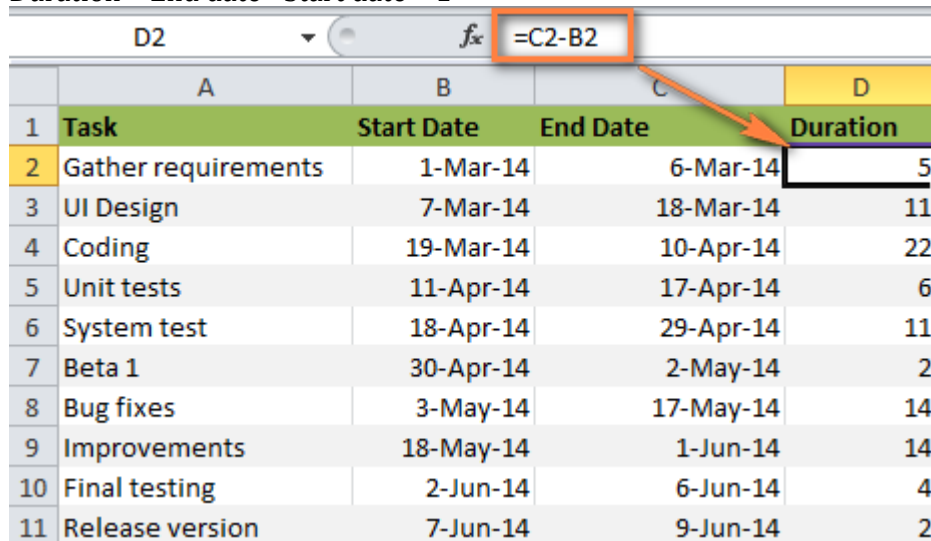
## 1. Create a project table

You start by entering your project's data in an Excel spreadsheet. List each task as a separate row and structure your project plan by including the *Start date*, *End date* and *Duration*, i.e. the number of days required to complete the tasks.

**Tip.** Only the *Start date* and *Duration* columns are necessary for creating an Excel Gantt chart. If you have *Start Dates* and *End Dates*, you can use one of these simple formulas to calculate *Duration*, whichever makes more sense for you:

Duration = End Date - Start Date

Duration = End date - Start date + 1

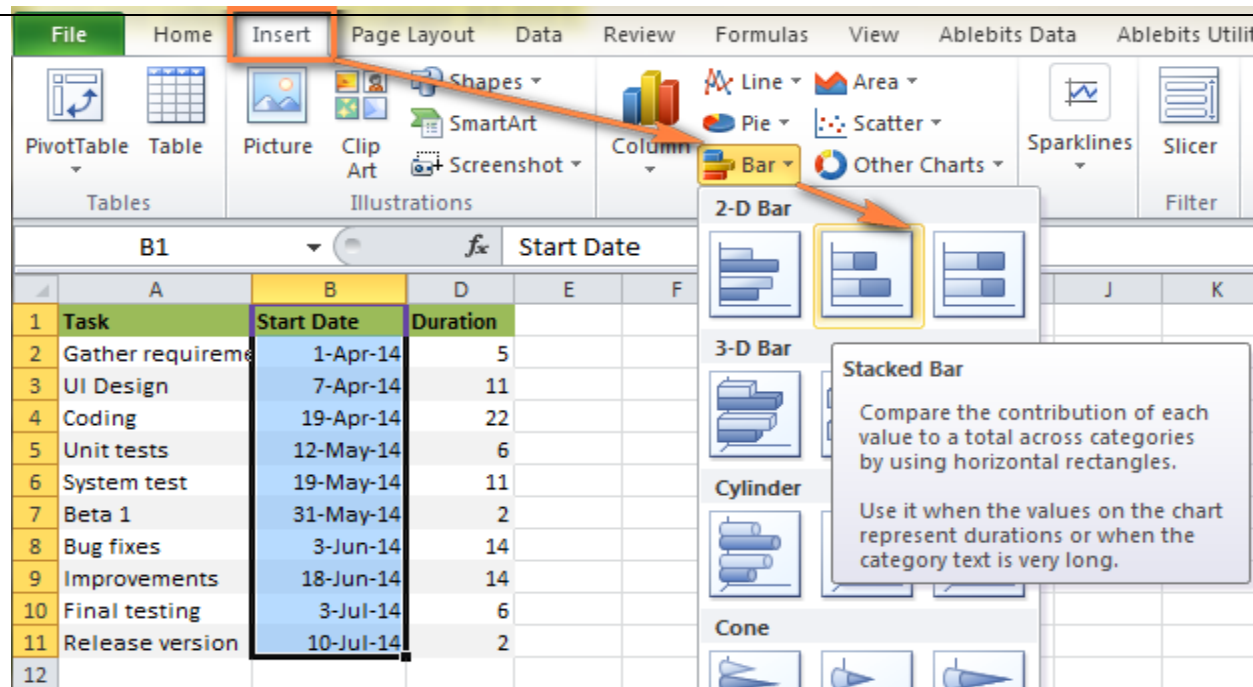


	A	B	C	D
1	Task	Start Date	End Date	Duration
2	Gather requirements	1-Mar-14	6-Mar-14	5
3	UI Design	7-Mar-14	18-Mar-14	11
4	Coding	19-Mar-14	10-Apr-14	22
5	Unit tests	11-Apr-14	17-Apr-14	6
6	System test	18-Apr-14	29-Apr-14	11
7	Beta 1	30-Apr-14	2-May-14	2
8	Bug fixes	3-May-14	17-May-14	14
9	Improvements	18-May-14	1-Jun-14	14
10	Final testing	2-Jun-14	6-Jun-14	4
11	Release version	7-Jun-14	9-Jun-14	2

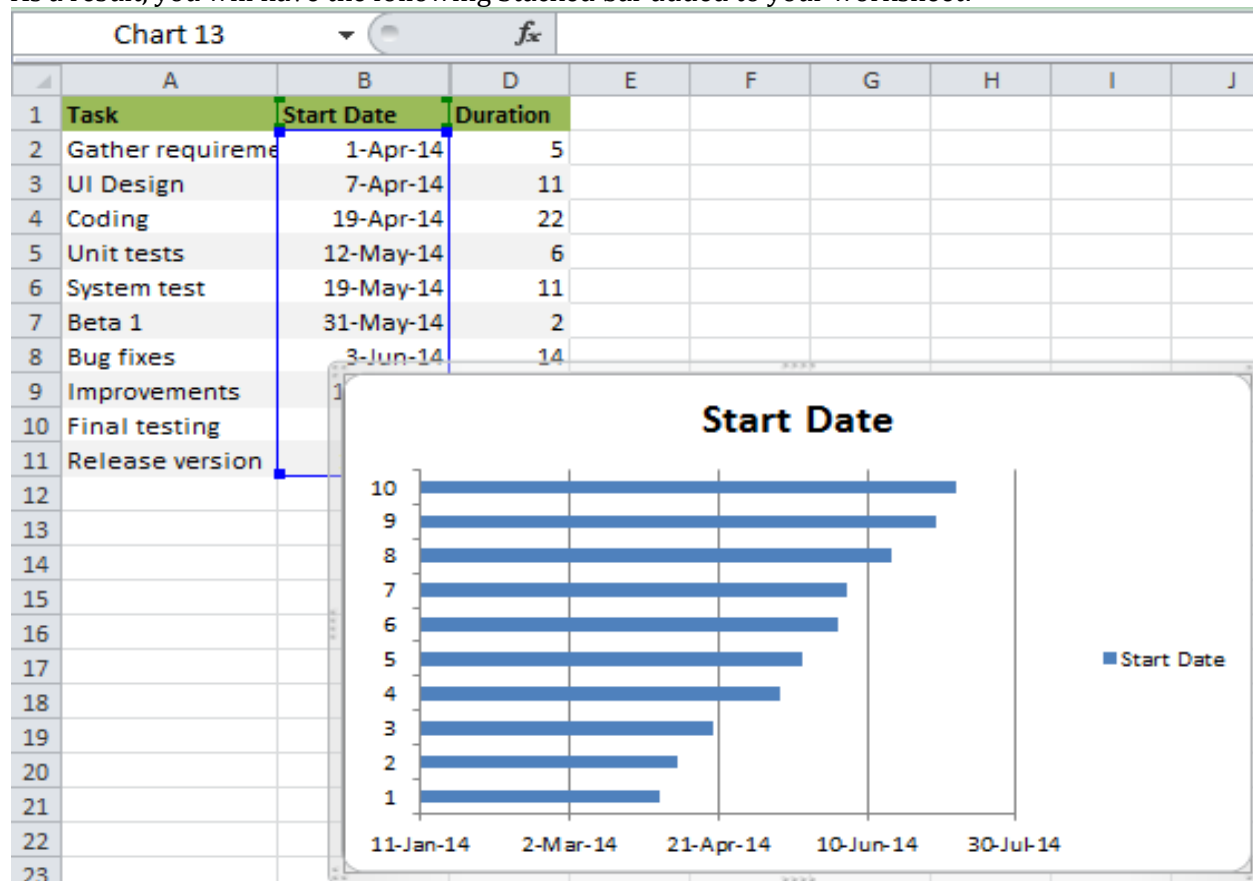
## 2. Make a standard Excel Bar chart based on Start date

You begin making your Gantt chart in Excel by setting up a usual *Stacked Bar* chart.

- Select a range of your **Start Dates** with the column header, it's B1:B11 in our case. Be sure to select only the cells with data, and not the entire column.
- Switch to the *Insert* tab > *Charts* group and click **Bar**.
- Under the *2-D Bar* section, click **Stacked Bar**.



As a result, you will have the following Stacked bar added to your worksheet:

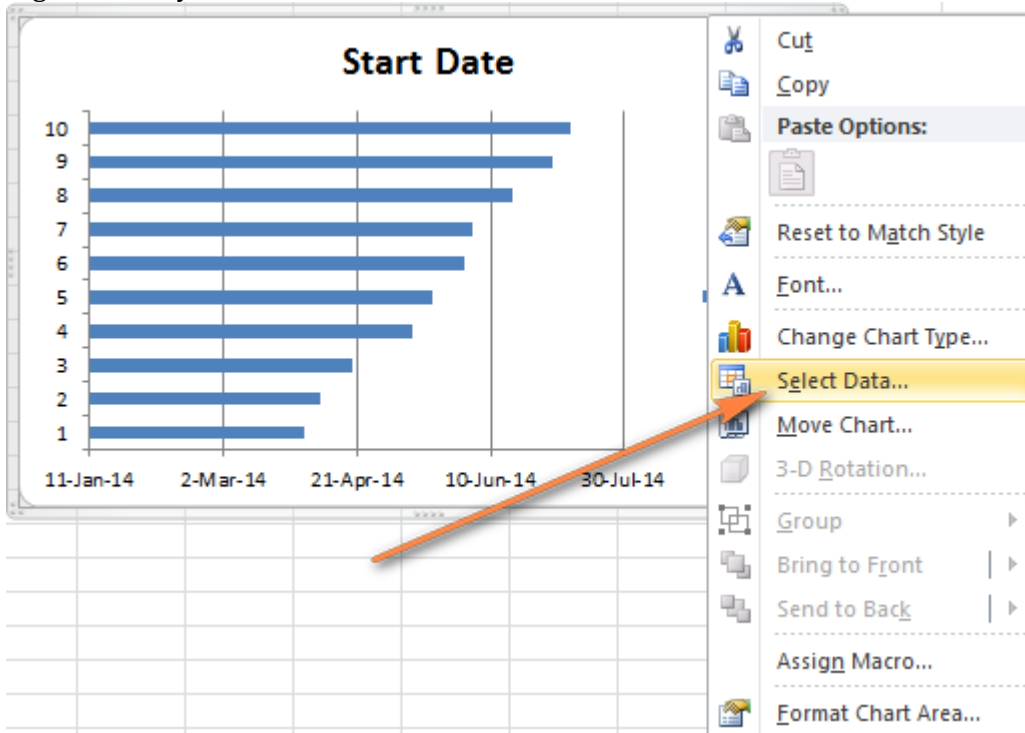


**Note.** Some other Gantt Chart tutorials you can find on the web recommend creating an empty bar chart first and then populating it with data as explained in the next step. But I think the above approach is better because Microsoft Excel will add one data series to the chart automatically, and in this way save you some time.

### 3. Add Duration data to the chart

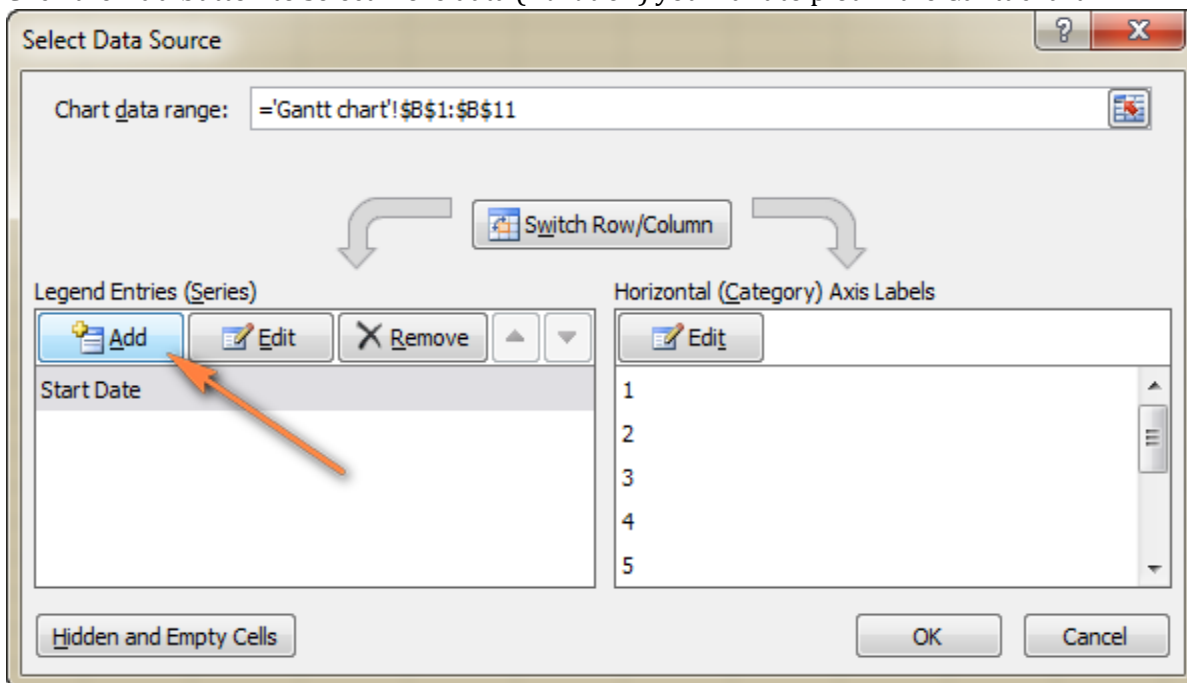
Now you need to add one more series to your Excel Gantt chart-to-be.


1. Right-click anywhere within the chart area and choose **Select Data** from the context menu.

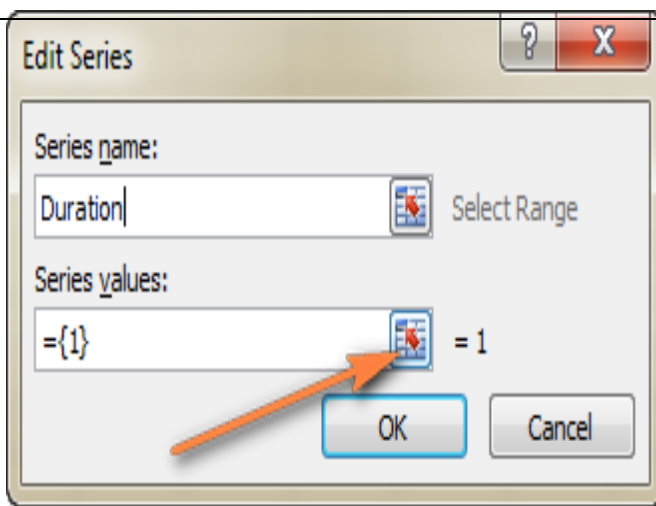


The **Select Data Source** window will open. As you can see in the screenshot below, *Start Date* is already added under **Legend Entries (Series)**. And you need to add *Duration* there as well.

2. Click the **Add** button to select more data (*Duration*) you want to plot in the Gantt chart.

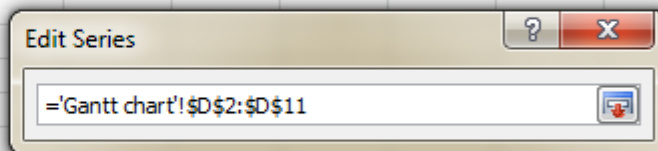


3. The *Edit Series* window opens and you do the following:
  - In the **Series name** field, type "*Duration*" or any other name of your choosing. Alternatively, you can place the mouse cursor into this field and click the column header in your spreadsheet, the clicked header will be added as the *Series name* for the Gantt chart.
  - Click the range selection icon  next to the **Series Values** field.

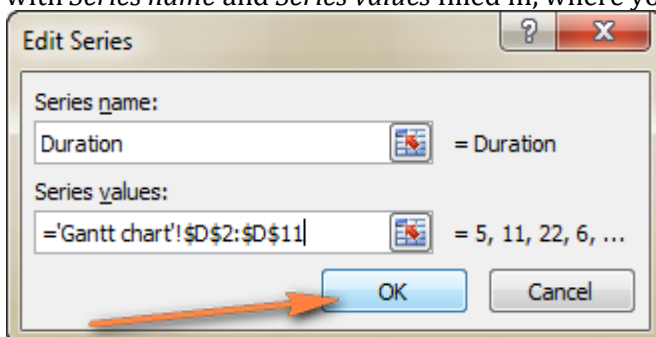


4. A small *Edit Series* window will open. Select your project **Duration** data by clicking on the first Duration cell (D2 in our case) and dragging the mouse down to the last duration (D11). Make sure you have not mistakenly included the header or any empty cell.

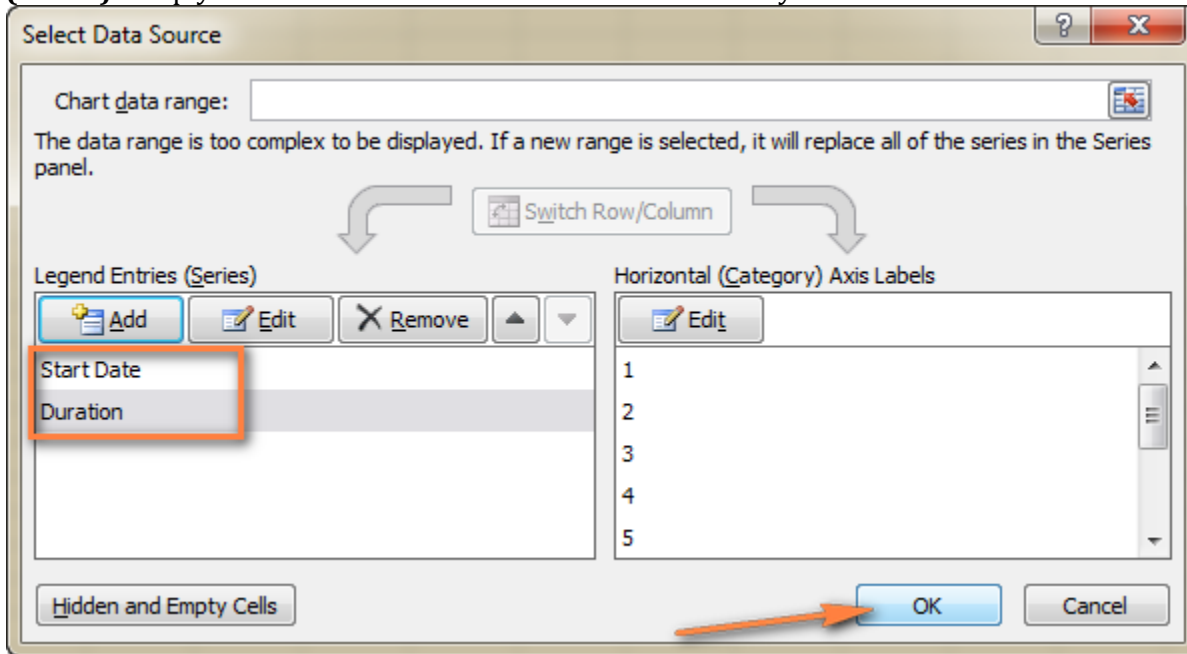
A	B	D	E	F	G	H	I	J	K
Task	Start Date	Duration							
Gather requirements	1-Apr-14	5							
UI Design	7-Apr-14	11							
Coding	19-Apr-14	22							
Unit tests	12-May-14	6							
System test	19-May-14	11							
Beta 1	31-May-14	2							
Bug fixes	3-Jun-14	14							
Improvements	18-Jun-14	14							
Final testing	3-Jul-14	6							
Release version	10-Jul-14	2							



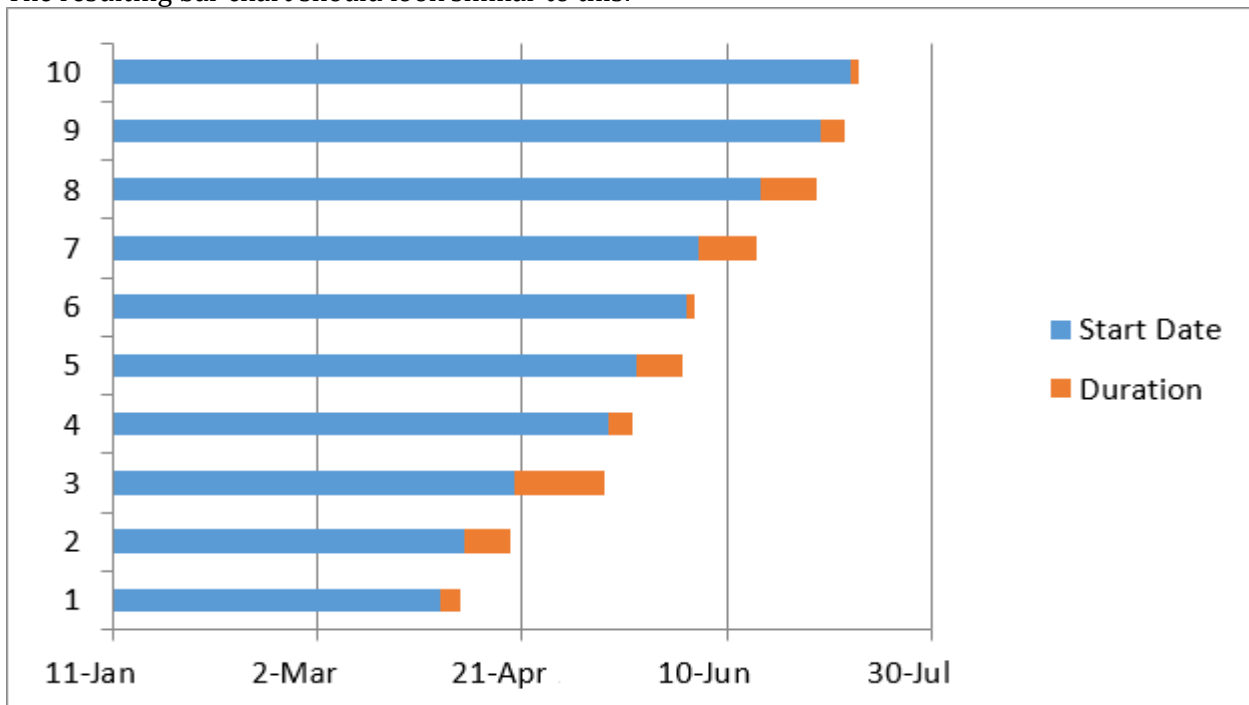
5. Click the Collapse Dialog icon to exit this small window. This will bring you back to the previous *Edit Series* window with *Series name* and *Series values* filled in, where you click *OK*.



6. Now you are back at the *Select Data Source* window with both *Start Date* and *Duration* added under **Legend Entries (Series)**. Simply click *OK* for the Duration data to be added to your Excel chart.



The resulting bar chart should look similar to this:

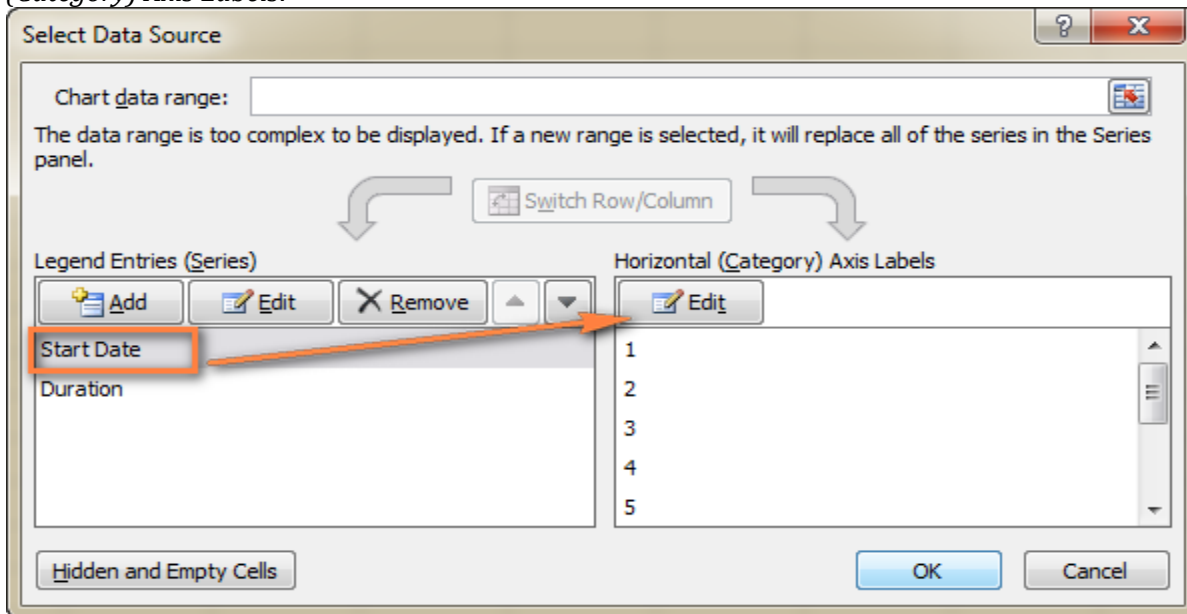



#### 4. Add task descriptions to the Gantt chart

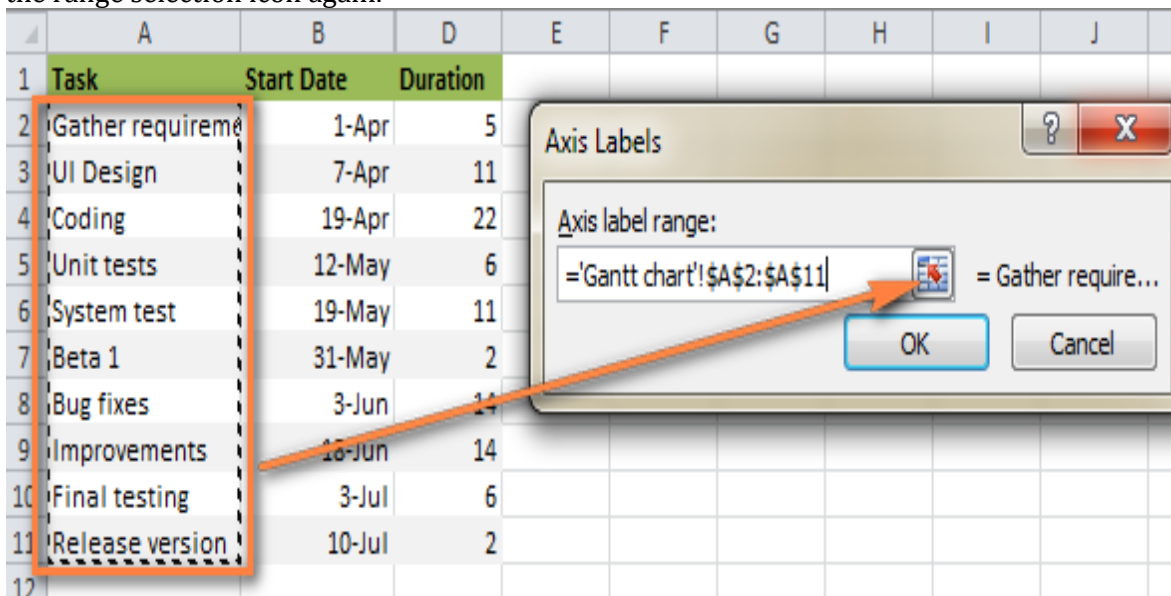
Now you need to replace the days on the left side of the chart with the list of tasks.

1. Right-click anywhere within the chart plot area (the area with blue and orange bars) and click **Select Data** to bring up the *Select Data Source* window again.

2. Make sure the **Start Date** is selected on the left pane and click the **Edit** button on the right pane, under *Horizontal (Category) Axis Labels*.

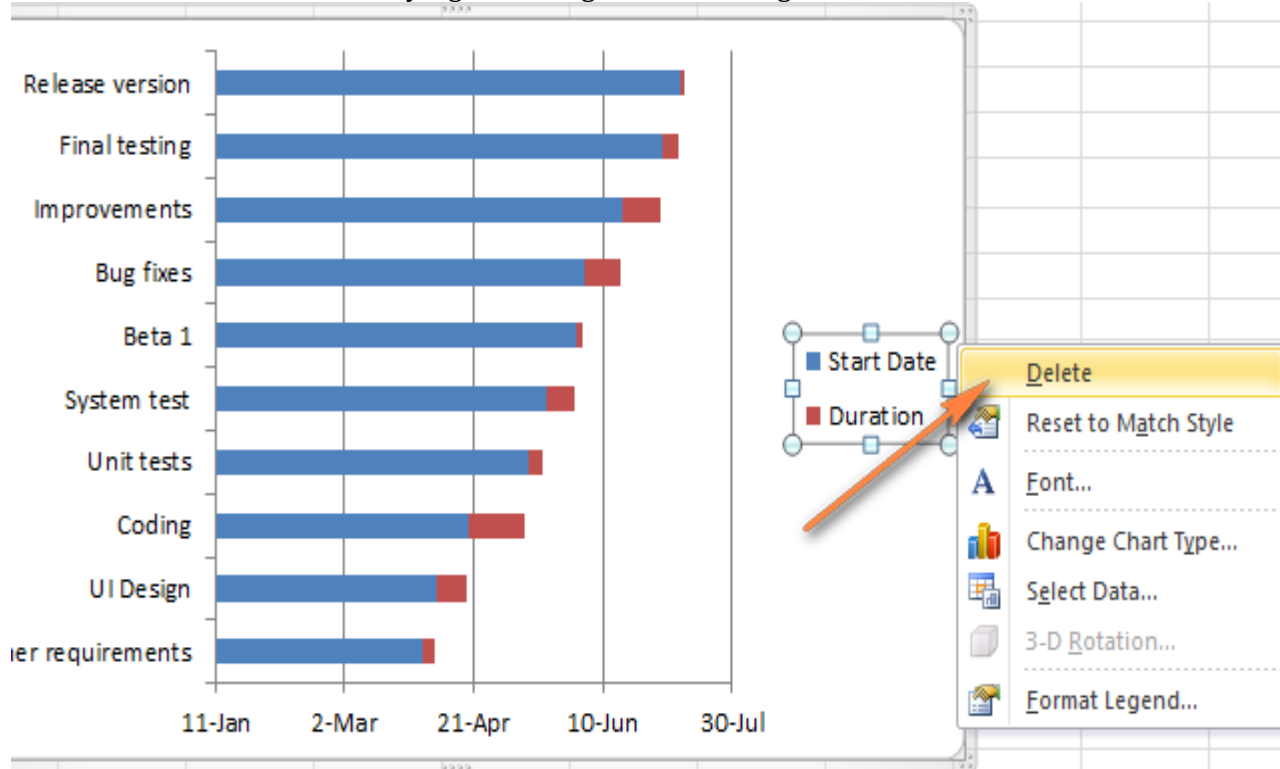


3. A small *Axis Label* window opens and you select your tasks in the same fashion as you selected Durations in the previous step - click the range selection icon , then click on the first task in your table and drag the mouse down to the last task. Remember, the column header should not be included. When done, exit the window by clicking on the range selection icon again.

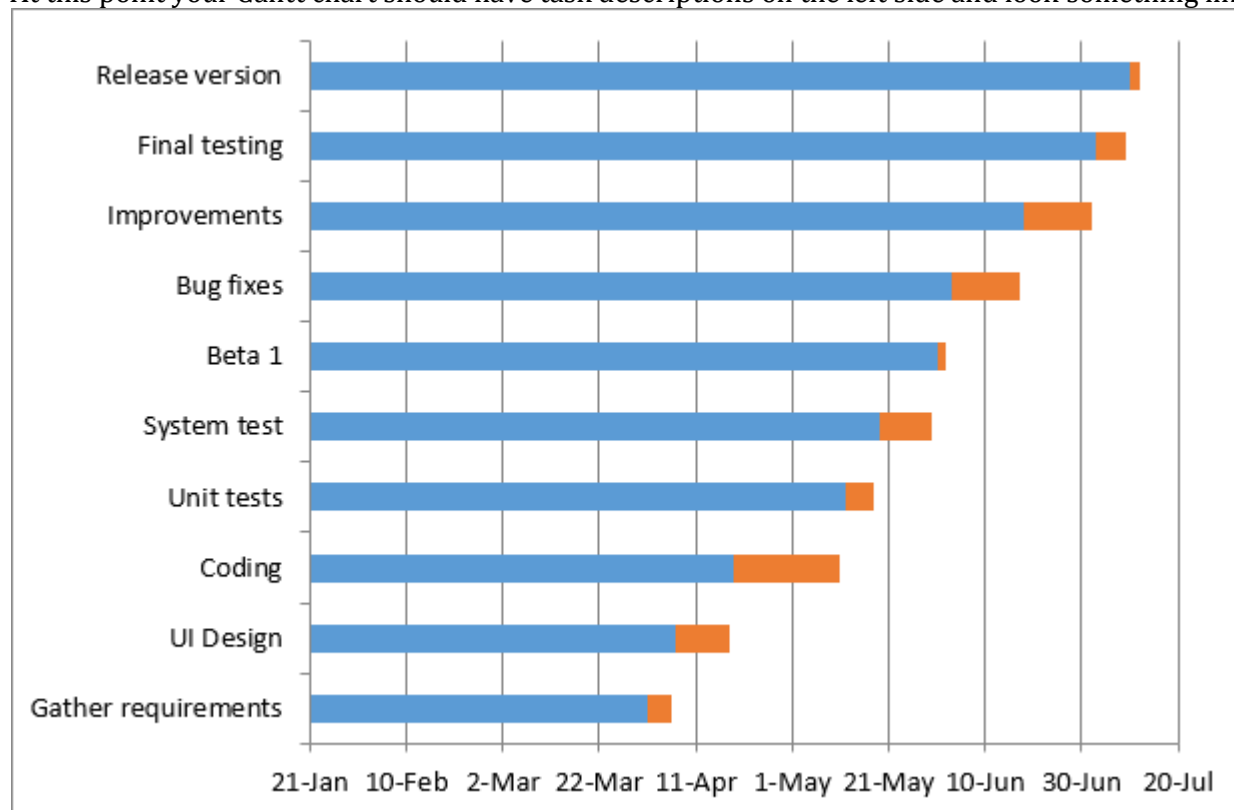


4. Click *OK* twice to close the open windows.

5. Remove the chart labels block by right-clicking it and selecting *Delete* from the context menu.



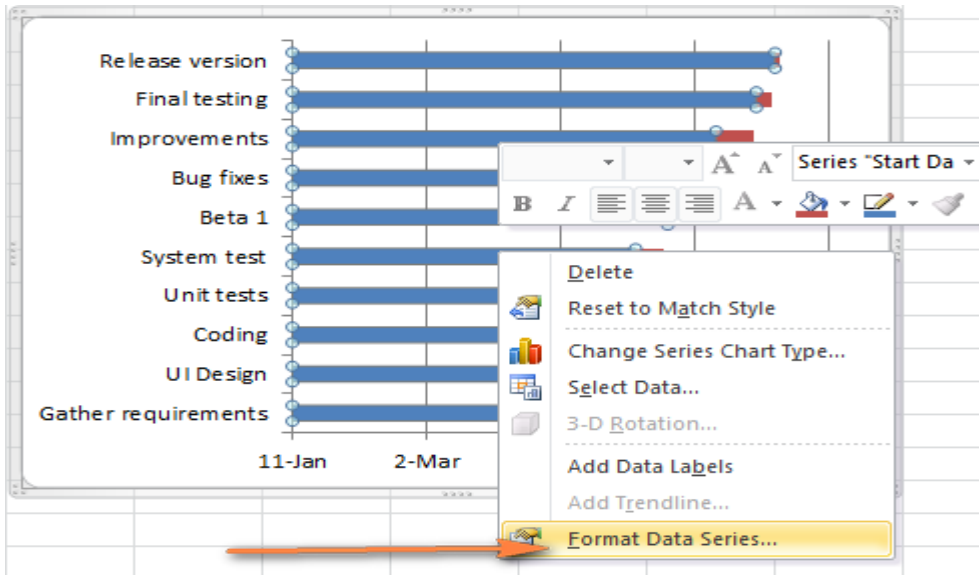
At this point your Gantt chart should have task descriptions on the left side and look something like this:



## 5. Transform the bar graph into the Excel Gantt chart

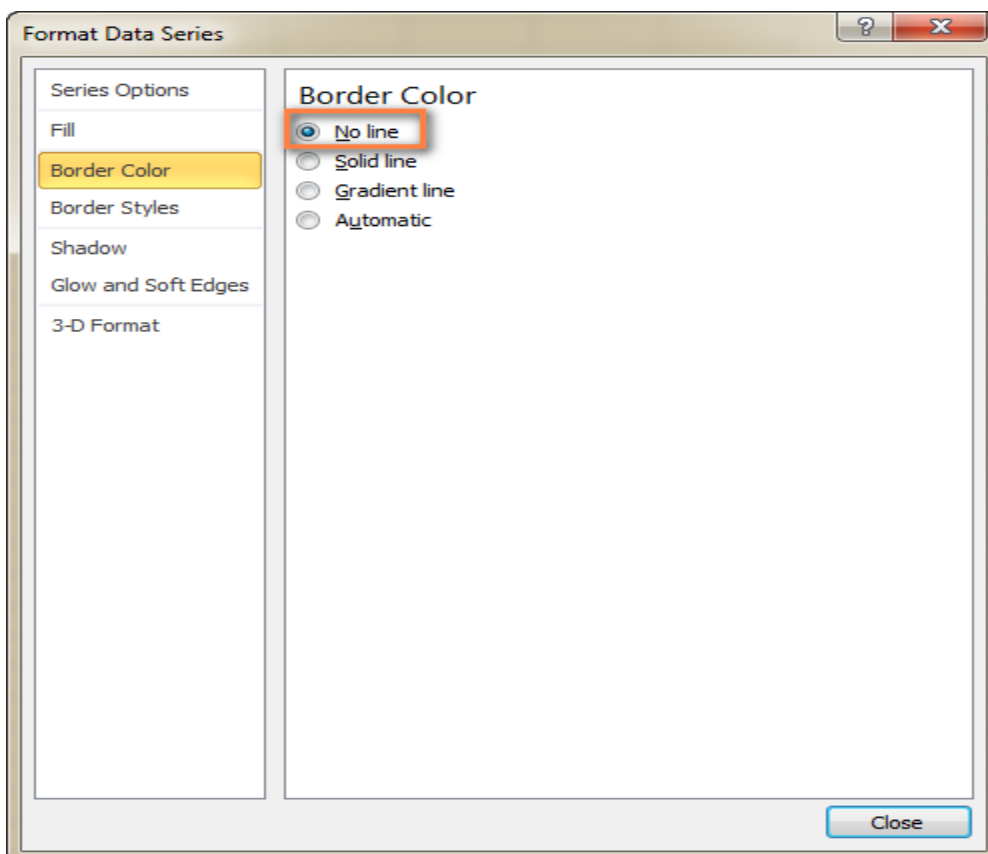
What you have now is still a stacked bar chart. You have to add the proper formatting to make it look more like a Gantt chart. Our goal is to remove the blue bars so that only the orange parts representing the project's tasks will be visible. In technical terms, we won't really delete the blue bars, but rather make them transparent and therefore invisible.

1. Click on any **blue bar** in your Gantt chart to select them all, right-click and choose **Format Data Series** from the context menu.



2. The *Format Data Series* window will show up and you do the following:

- Switch to the *Fill* tab and select **No Fill**.
- Go to the *Border Color* tab and select **No Line**.

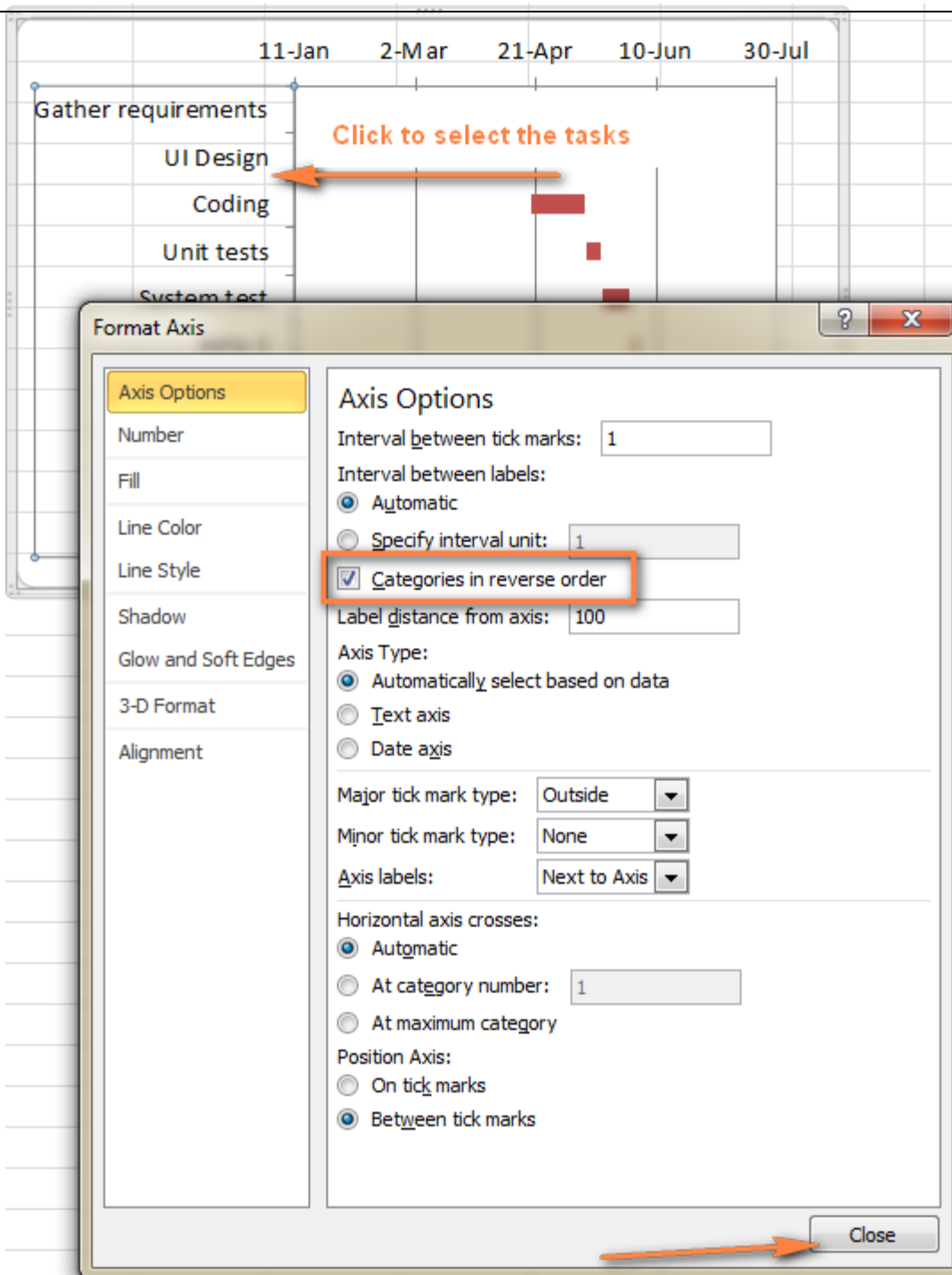


**Note.** You do not need to close the dialog because you will use it again in the next step.

3. As you have probably noticed, the tasks on your Excel Gantt chart are listed in **reverse order**. And now we are going to fix this.

Click on the list of tasks in the left-hand part of your Gantt chart to select them. This will display the *Format Axis* dialog for you. Select the **Categories in reverse order** option under *Axis Options* and then click the *Close* button to save all the changes.

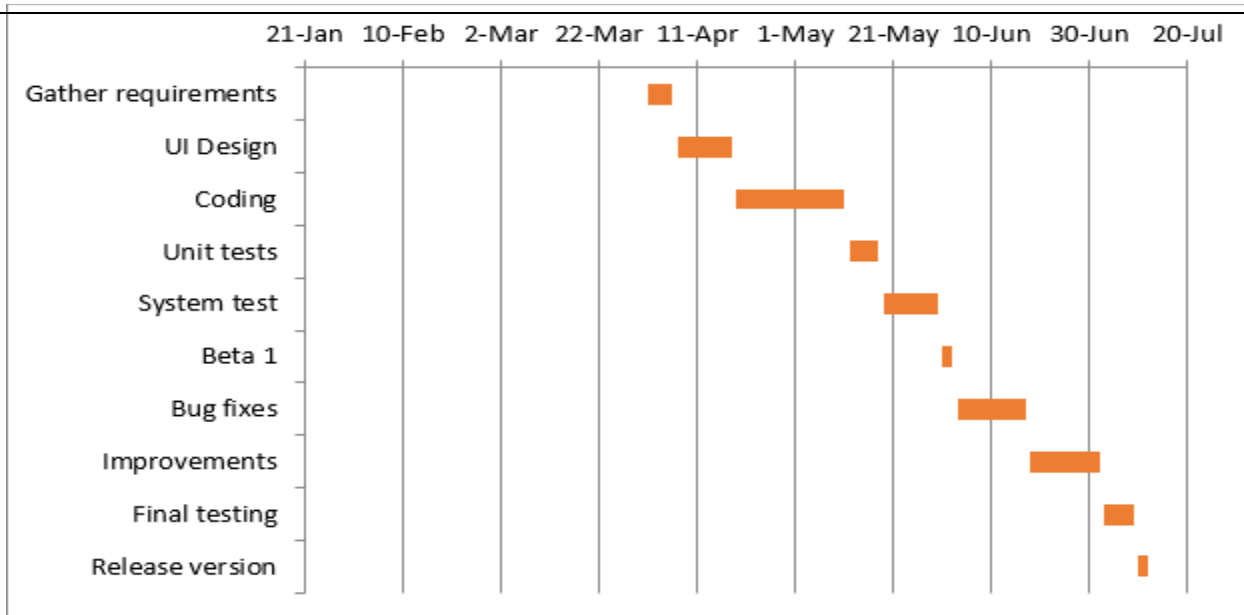




The results of the changes you have just made are:

- Your tasks are arranged in a proper order on a Gantt chart.
- Date markers are moved from the bottom to the top of the graph.

Your Excel chart is starting to look like a normal Gantt chart, isn't it? For example, my Gantt diagram looks like this now:



## 6. Improve the design of your Excel Gantt chart

Though your Excel Gantt chart is beginning to take shape, you can add a few more finishing touches to make it really stylish.

### 1. Remove the empty space on the left side of the Gantt chart.

As you remember, originally the starting date blue bars resided at the start of your Excel Gantt diagram. Now you can remove that blank white space to bring your tasks a little closer to the left vertical axis.

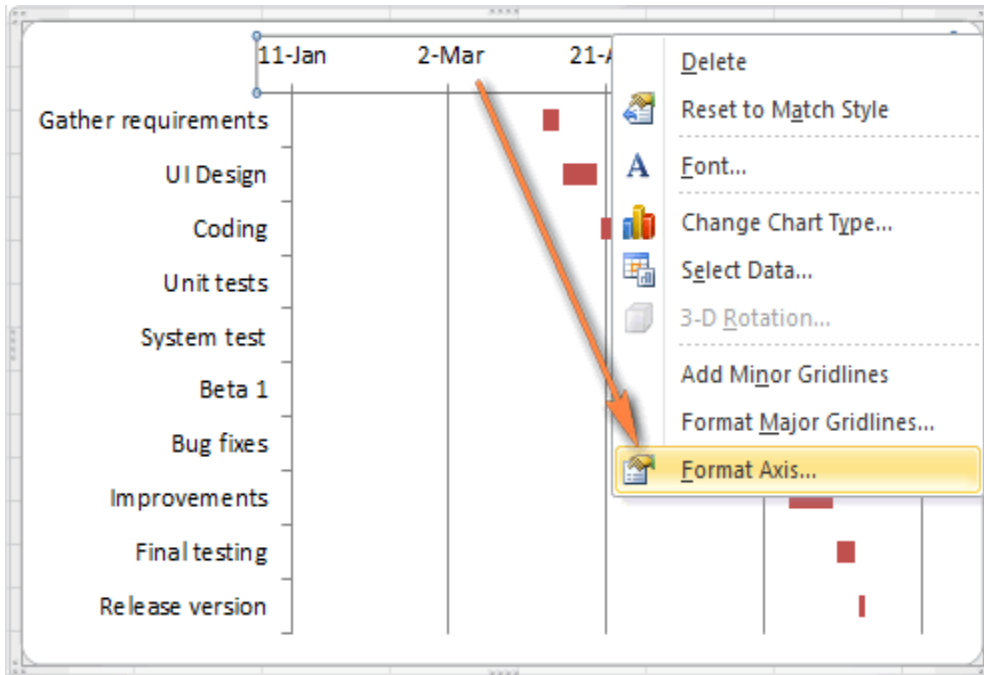
- Right-click on the **first Start Date** in your data table, select *Format Cells > General*. Write down the number that you see - this is a numeric representation of the date, in my case 41730. As you probably know, Excel stores dates as numbers based on the number of days since 1-Jan-1900. Click *Cancel* because you don't actually want to make any changes here.

Task	Start Date	Duration
Gather requirements	1-Apr	5
UI Design	7-Apr	11
Coding	19-Apr	22
Unit tests	12-May	6

Format Cells dialog box (General tab):

- Category: General
- Sample: 41730
- General format cells have no specific number format.

- Click on any date above the task bars in your Gantt chart. One click will select all the dates, you right click them and choose **Format Axis** from the context menu.



- Under *Axis Options*, change **Minimum** to **Fixed** and type the number you recorded in the previous step.

## 2. Adjust the number of dates on your Gantt chart.

In the same *Format Axis* window that you used in the previous step, change **Major unit** and **Minor unit** to **Fixed** too, and then add the numbers you want for the date intervals. Typically, the shorter your project's timeframe is, the smaller numbers you use. For example, if you want to show every other date, enter 2 in the *Major unit*. You can see my settings in the screenshot below.

**Note.** In Excel 365, Excel 2021 - 2013, there are no *Auto* and *Fixed* radio buttons, so you simply type the number in the box.

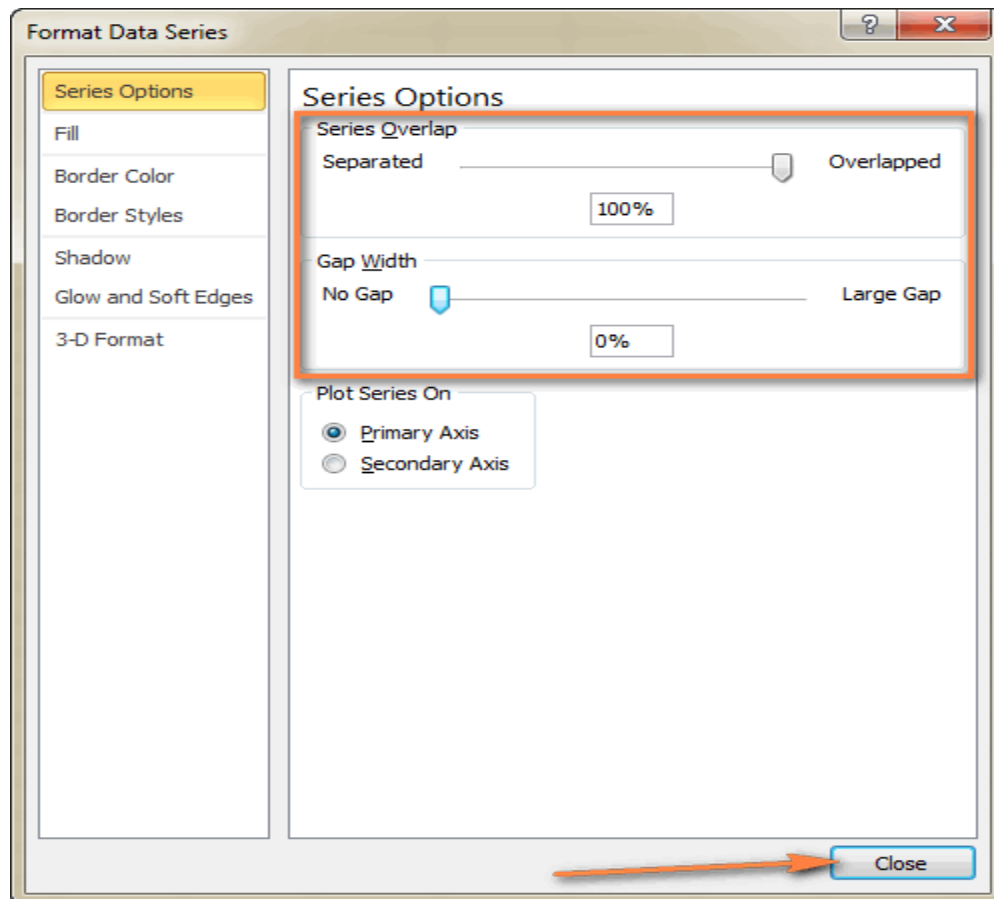
**Tip.** You can play with different settings until you get the result that works best for you. Don't be afraid to do something wrong because you can always revert to the default settings by switching back to *Auto* in Excel 2010 and

2007, or click *Reset* in Excel 2013 and later.

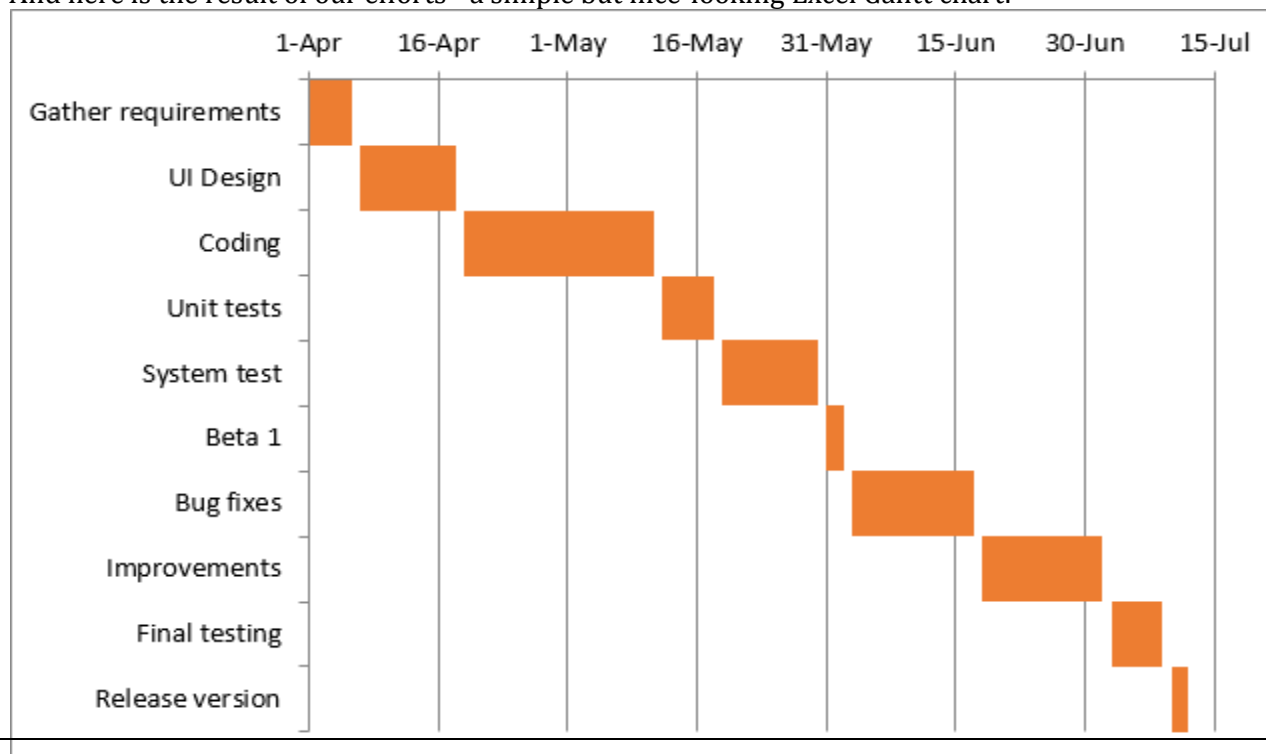
### 3. Remove excess white space between the bars.

Compacting the task bars will make your Gantt graph look even better.

- Click any of the orange bars to get them all selected, right click and select **Format Data Series**.
- In the Format Data Series dialog, set **Separated** to **100%** and **Gap Width** to **0%** (or close to 0%).



And here is the result of our efforts - a simple but nice-looking Excel Gantt chart:

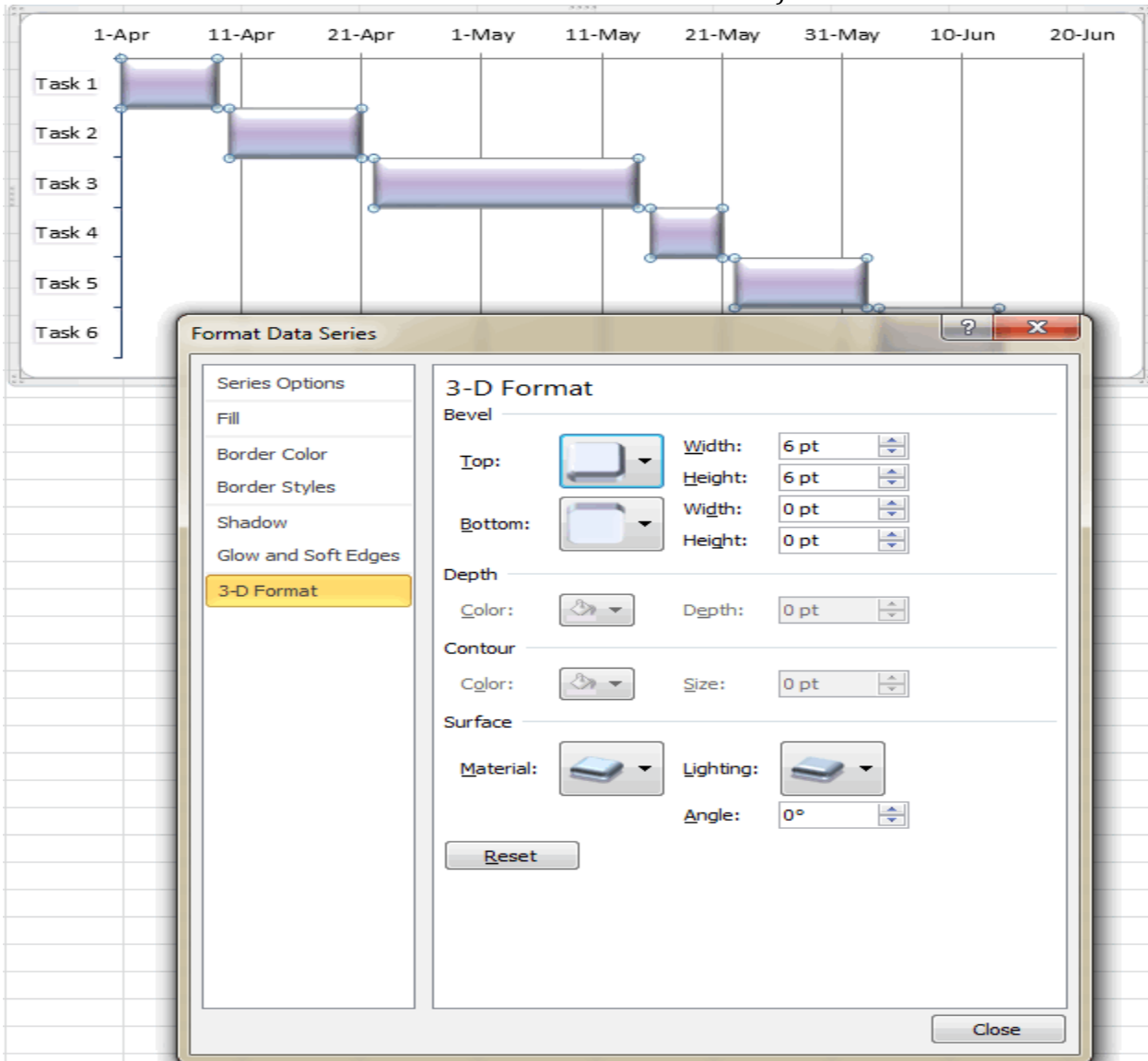


Remember, though your Excel chart simulates a Gantt diagram very closely, it still keeps the main features of a standard Excel chart:

- Your Excel Gantt chart will resize when you add or remove tasks.
- You can change a Start date or Duration, the chart will reflect the changes and adjust automatically.
- You can [save your Excel Gantt chart as an image](#) or [convert to HTML and publish online](#).

### Tips:

- You can design your Excel Gantt chart in different ways by changing the fill color, border color, shadow and even applying the 3-D format. All these options are available in the **Format Data Series** window (right-click the bars in the chart area and select *Format Data Series* from the context menu).



- When you have created an awesome design, it might be a good idea to save your Excel Gantt chart as a template for future use. To do this, click the chart, switch to the *Design* tab on the ribbon and click **Save as Template**.

## ASSIGNMENT NO: 07

### Title: Generate Dashboard and Reports

- **Dashboard**
  - Project Overview
  - Cost Overview
  - Upcoming Tasks
- **Resource Reports**
  - Over-allocated Resources
  - Resource Overview
- **Cost Reports**
  - Earned Value Report
  - Resource Cost Overview
  - Task Cost Overview
- **Progress Reports**
  - Critical Tasks
  - Milestone Report
  - Slipping Tasks

### Theory:

#### What is a project dashboard?

A project dashboard is a [project management](#) tool businesses use to track key performance indicators for various projects. Dashboards can show performance metrics, display progress reports and highlight areas that require attention. They can help monitor the success of specific campaigns, processes and projects. Regardless of a company's size, it likely can benefit from using a project dashboard.

Besides tracking performance, project dashboards can also be helpful communication and organizational tools that assist companies in monitoring budgets, timelines and tasks. They make it easier to measure results and manage multiple moving parts simultaneously. Some common metrics these dashboards measure include:

- Task status
- Project progress
- Hours per task
- Risks and changes
- Billable and unbillable hours
- Budget
- Total revenue

#### Features of a project dashboard

Many features in a project dashboard help to enhance organization, improve tracking and increase company and employee evaluation capabilities. They also provide a central location where companies can easily access all their data, metrics and timeline information. This can make managing multiple project steps, service offerings or large teams easier and more efficient. The capabilities are highly customizable, meaning you can adjust your dashboard to meet your specific needs.

For example, marketing teams may use dashboards to track various campaigns. Developers may use them to track a project's progress or record the number of [billable hours](#) they've spent on a project. Whatever your goals, there's probably a dashboard that can provide useful and relevant metrics for your business. Consider finding dashboards that are user-friendly, customizable and flexible so that you can ensure you have the best fit for your needs. Some features you can add to optimize your dashboard include:

- **Project roadmap:** This plan includes the tasks a team will complete during a project and what features they will add to the product or service they're creating. A project roadmap also includes who's responsible for which task, which can prevent miscommunications.
- **Financial widget:** This feature might be best as a graph, and it can show your budget for the project and how you plan to [allocate resources](#) for it. This can help you compare how much your team spends on the project to how much you budgeted.
- **Task tracker:** This widget can change to show what tasks are the main priorities that day. You also can use this feature to label what team or team member is completing the task.
- **Real-time updates:** Creating your dashboard to update in real-time can help ensure your team uses the most accurate and relevant data when working on a project. Automating this process also can help your team save time, allowing them to devote more time to work on tasks.

## A. Plan the Project

### A1. Scope Management Plan

- ☒ The project scope is clearly defined
- ☒ The project scope is consistent with the initial scope as defined in the Project Charter
- ☒ It is clear what is in scope and out of scope of the project
- ☒ The scope has been baselined and is clearly defined in the Work Breakdown Structure
- ☒ There is a clearly defined process for the management of the project scope
- ☒ Roles and responsibilities for scope management have been assigned and are clear
- ☒ Tailoring, as well as scope management tools and techniques, are described
- ☒ Project assumptions have been identified
- ☐ Project constraints have been identified
- ☒ Project dependencies have been identified

### A2. Schedule Management Plan

- ☒ The schedule has been baselined and key milestones have been identified
- ☐ The project estimate method has been identified
- ☒ There is a clearly defined process for the management of the project schedule
- ☒ Roles and responsibilities for schedule management have been assigned and are clear
- ☒ Tailoring, as well as schedule management tools and techniques, are described

### A3. Cost Management Plan

- ☒ The project cost has been baselined and cost items have been identified
- ☐ There is a clearly defined process for invoicing in the project
- ☒ There is a clearly defined process for the management of costs in the project
- ☒ Roles and responsibilities for cost management have been assigned and are clear
- ☒ Tailoring, as well as cost management tools and techniques, are described

### A4. Quality Management Plan

- ☒ A definition of quality has been agreed for the project

- ☒ Acceptance criteria have been identified and agreed for the project
- ☐ A quality assurance plan, with reviews, has been established and baselined
- ☒ There is a clearly defined process for the management of quality in the project
- ☒ Roles and responsibilities for quality management have been assigned and are clear
- ☒ Tailoring, as well as quality management tools and techniques, are described

#### **A5. Human Resources Management Plan**

- ☒ The project governance structure has been defined and agreed
- ☒ The project management team is clearly defined
- ☒ There is a clearly defined approach for staff acquisition, development, and transition
- ☒ There is a clearly defined process for the management of human resources in the project
- ☒ Roles and responsibilities for resource management have been assigned and are clear
- ☒ Tailoring, as well as resource management tools and techniques, are described

#### **A6. Communication Management Plan**

- ☒ Communication requirements have been identified
- ☒ A Communication Matrix has been defined
- ☒ The frequency, participants, and purpose of project meetings and reports is clearly defined
- ☒ There is a clearly defined process for the management of communications in the project
- ☒ Roles and responsibilities for communication management have been assigned and are clear
- ☒ Tailoring, as well as communication management tools and techniques, are described

#### **A7. Stakeholder Management Plan**

- ☒ The project stakeholders have been clearly identified
- ☒ There are clearly defined strategies for stakeholder engagement
- ☒ There is a clearly defined process for the management of stakeholders in the project
- ☒ Roles and responsibilities for stakeholder management have been assigned and are clear
- ☐ Tailoring, as well as stakeholder management tools and techniques, are described

#### **A8. Risk Management Plan**

- ☒ A risk register has been produced, with clear ownership and defined mitigation responses
- ☒ Scales for risk assessment have been clearly defined
- ☐ A risk contingency budget has been provisioned
- ☒ There is a clearly defined process for the management of risks in the project
- ☒ Roles and responsibilities for risk management have been assigned and are clear
- ☒ An issue register has been produced, with clear ownership and defined resolution strategies
- ☒ Scales for issue priority have been clearly defined
- ☒ There is a clearly defined process for the management of issues in the project



- ☒ Roles and responsibilities for issue management have been assigned and are clear
- ☒ Escalation routes and thresholds have been defined for risks and issues
- ☒ Tailoring, as well as risk and issue management tools and techniques, are described

#### **A9. Configuration Management Plan**

- ☒ Configuration items are clearly defined and have been baselined
- ☐ Configuration audits have been planned for the project
- ☒ There is a clearly defined process for the management of configuration in the project
- ☒ Roles and responsibilities for configuration management have been assigned and are clear
- ☒ Tailoring, as well as configuration management tools and techniques, are described

#### **A10. Procurement Management Plan**

- ☒ A contract type has been selected as appropriate for the project
- ☒ Procurement risks have been identified, have clear ownership, and mitigation responses
- ☒ Contract decision criteria have been clearly defined
- ☒ Performance metrics for vendors have been identified
- ☒ There is a clearly defined process for the project contract award
- ☒ There is a clearly defined process for the management of procurement in the project
- ☒ Roles and responsibilities for procurement management have been assigned and are clear
- ☒ Tailoring, as well as procurement management tools and techniques, are described

#### **B. Project Management Plan**

- ☒ Ownership of the Project Management Plan is clearly defined
- ☒ All subsidiary management plans have been assembled in the Project Management Plan
- ☒ Referenced documents are clearly identified
- ☒ There is a clearly defined process for the management of changes of the Project Management Plan
- ☒ The Project Management has been circulated to the project team
- ☐ The Project Management Plan has been baselined and signed-off by the Project Board