Tutorial-9: Parameters and Stack Frames

AbdulWahab Kabani

1

Objectives

- Review Passing Parameters to a subroutine
- Difference Between Passing by value and by reference
- Defining a dynamic workspace for a subroutine (stack frame).

2

Outline

- Passing Parameters
- Value vs. Reference
- Passing By Value (revision)
- Passing By Reference
- Stack Frame
- Conclusion

3

Outline

- **Passing Parameters**
- Value vs. Reference
- Passing By Value (revision)
- Passing By Reference
- Stack Frame
- Conclusion

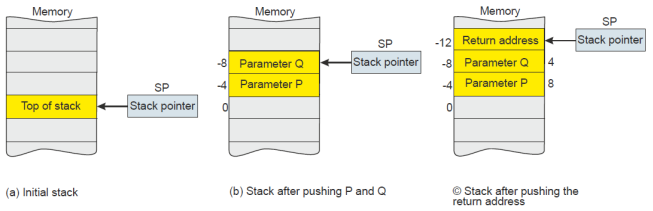
4

Passing Parameters

- **Question:** How do we pass parameters to a subroutine?

5

Passing Parameters



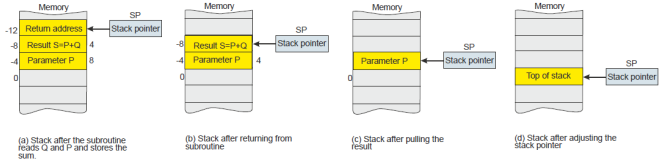
(a) Initial stack

(b) Stack after pushing P and Q

© Stack after pushing the return address

6

Passing Parameters



(a) Stack after the subroutine reads Q and P and stores the sum.

(b) Stack after returning from subroutine

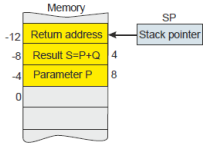
(c) Stack after pulling the result

(d) Stack after adjusting the stack pointer

7

Passing Parameters

- Steps:
  1. In the caller, push parameters to the stack
  2. In the caller, Push Return Address
  3. In the subroutine, calculate subroutine result
  4. In the subroutine, return the value
  5. In the subroutine, change the value of PC (return to call)



(a) Stack after the subroutine reads Q and P and stores the sum.

8

Outline

- **Passing Parameters**
- **Value vs. Reference**
- Passing By Value (revision)
- Passing By Reference
- Stack Frame
- Conclusion

9

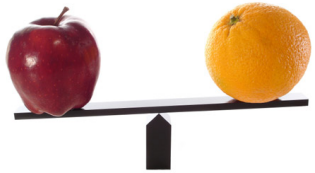
Value vs. Reference

- **Question**: what is the difference between passing parameters by value and passing by reference?

10

Value vs. Reference

- Passing By Value:
  - anything passed into a function call is unchanged in the caller's scope
- Passing By Reference:
  - A reference (memory address) to a variable is passed
  - The function can modify the parameter



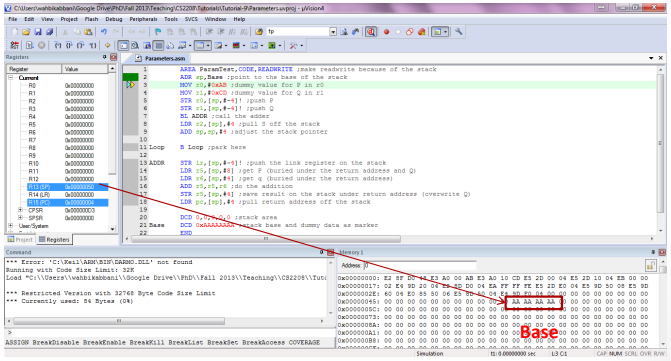
11

Outline

- **Passing Parameters**
- **Value vs. Reference**
- **Passing By Value (revision)**
- Passing By Reference
- Stack Frame
- Conclusion

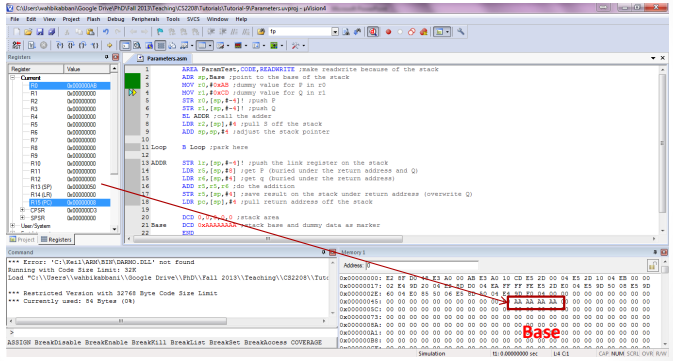
12

Passing By Value



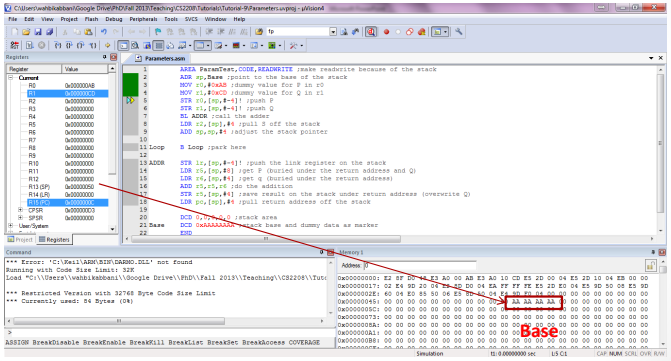
13

Passing By Value



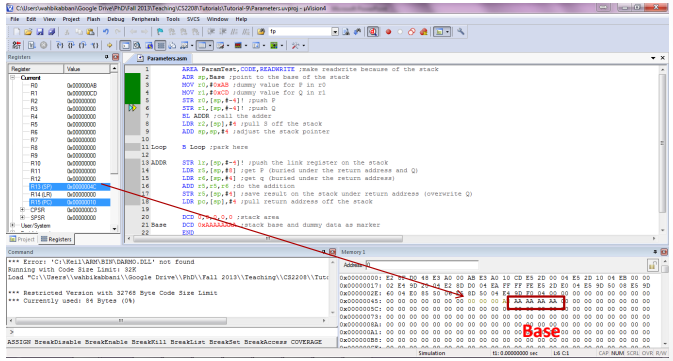
14

Passing By Value



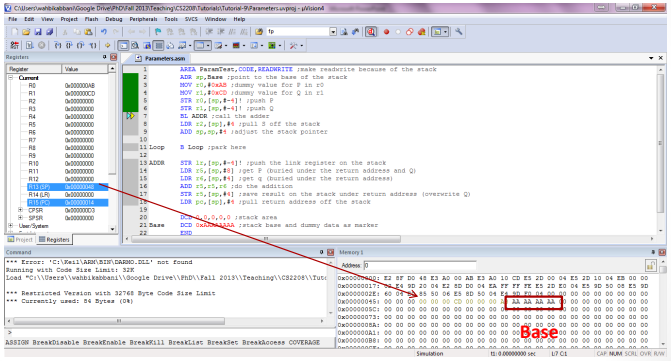
15

Passing By Value



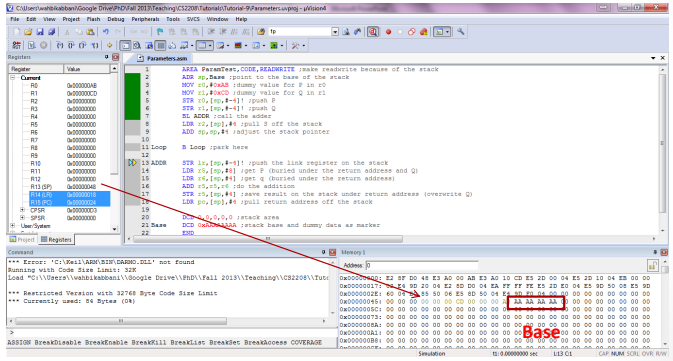
16

Passing By Value



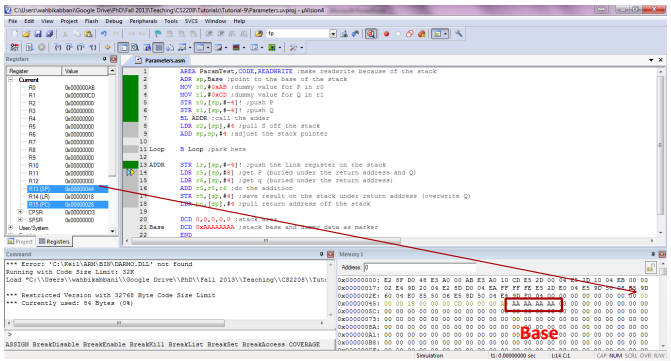
17

Passing By Value



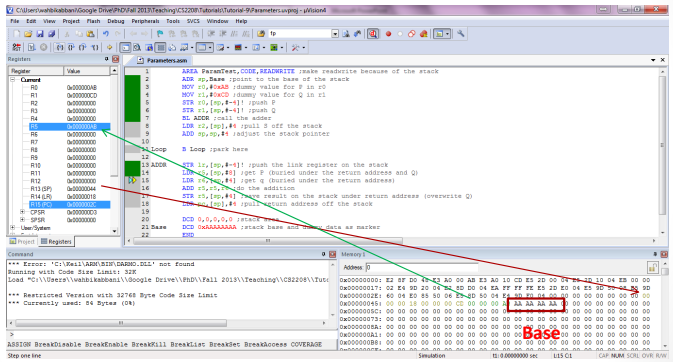
18

Passing By Value



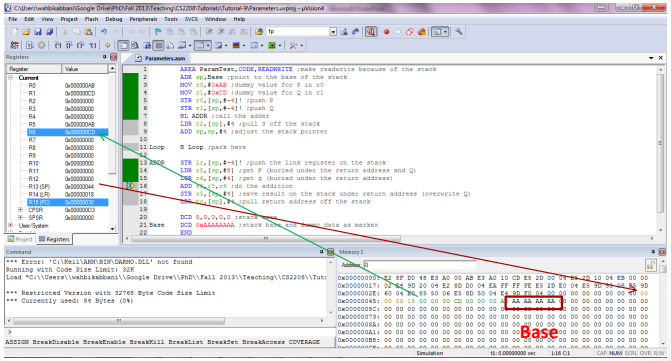
19

Passing By Value



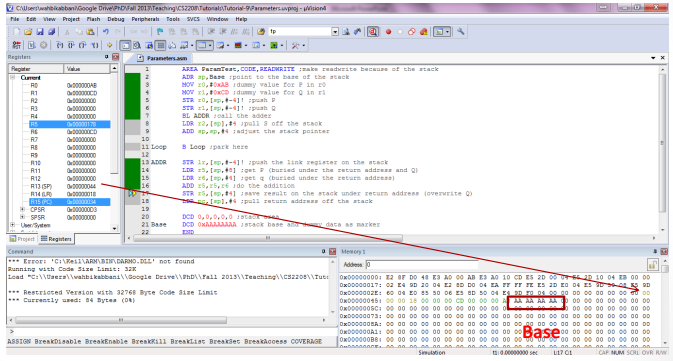
20

Passing By Value



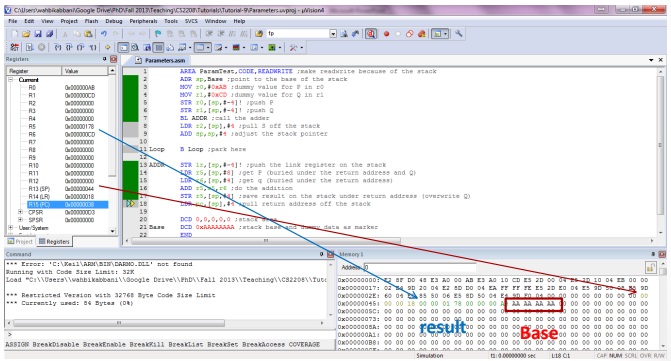
21

Passing By Value



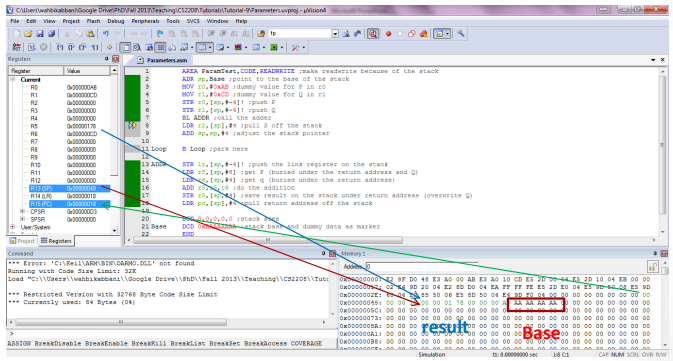
22

Passing By Value



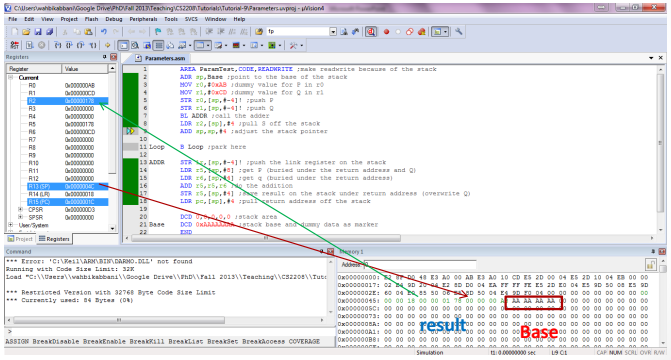
23

Passing By Value



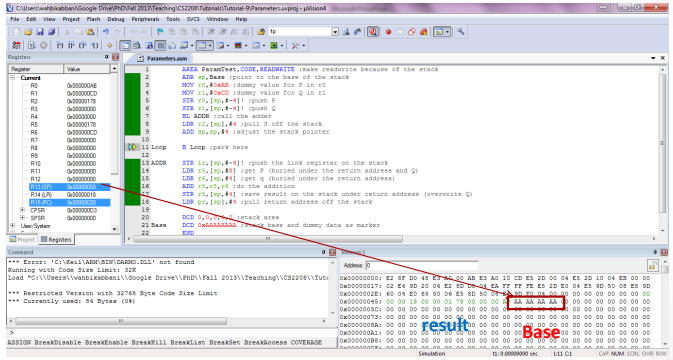
24

Passing By Value



25

Passing By Value



26

Outline

- Passing Parameters
- Value vs. Reference
- Passing By Value (revision)
- Passing By Reference
- Stack Frame
- Conclusion

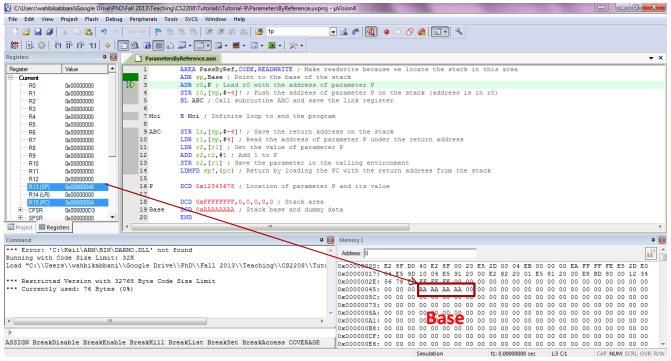
27

Passing By Reference

- The function can modify/change the parameter passed to it
- Can be done by passing a reference (memory address) to the parameter

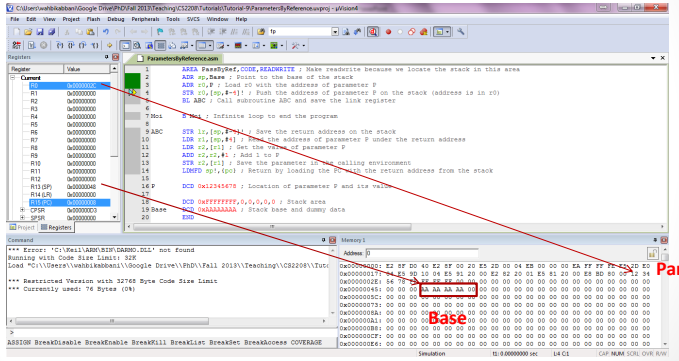
28

Passing By Reference



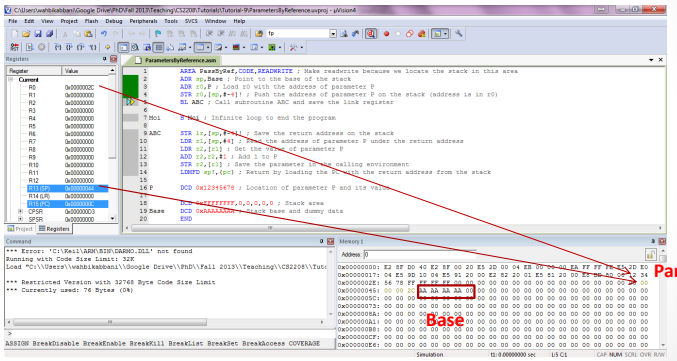
29

Passing By Reference



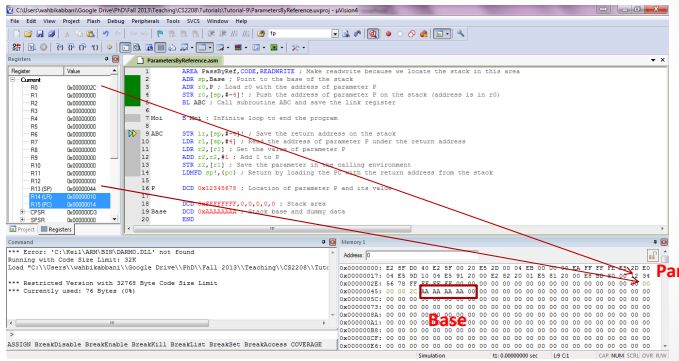
30

Passing By Reference



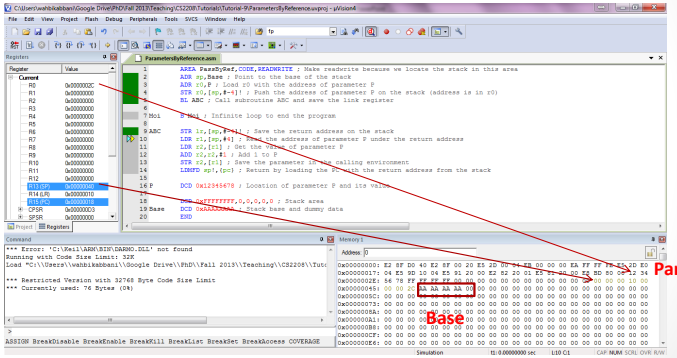
31

Passing By Reference



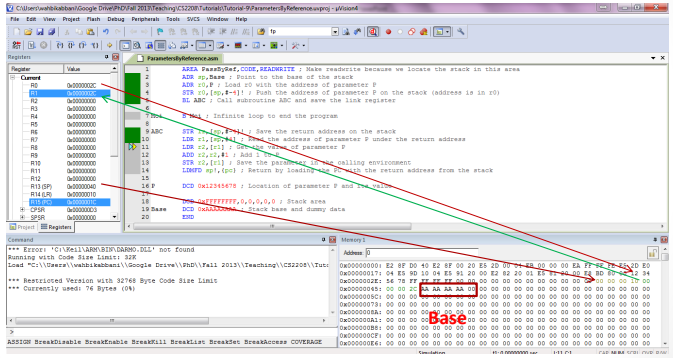
32

Passing By Reference



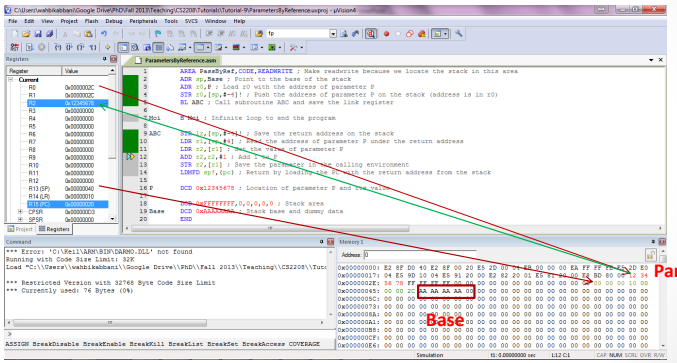
33

Passing By Reference



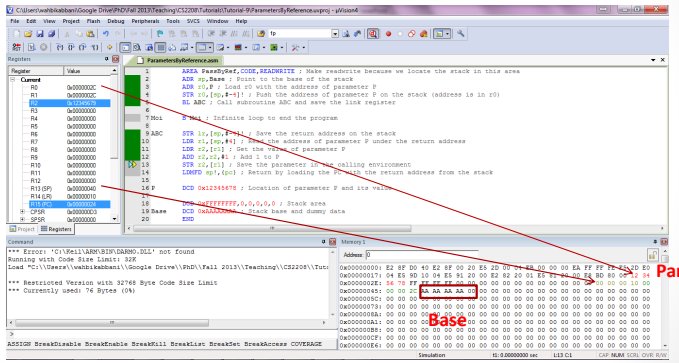
34

Passing By Reference



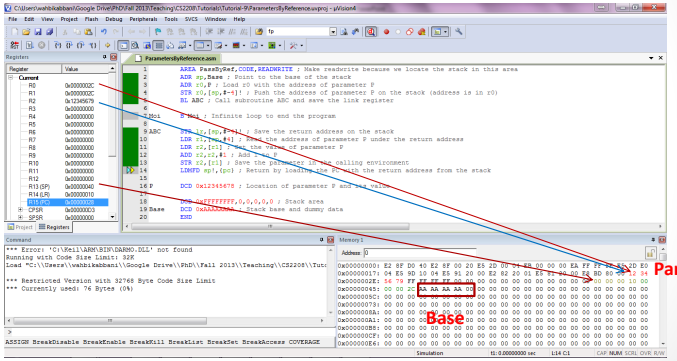
35

Passing By Reference



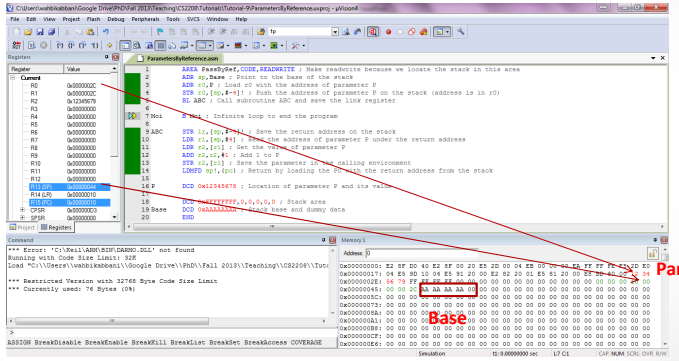
36

Passing By Reference



37

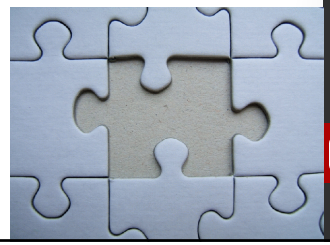
Passing By Reference



38

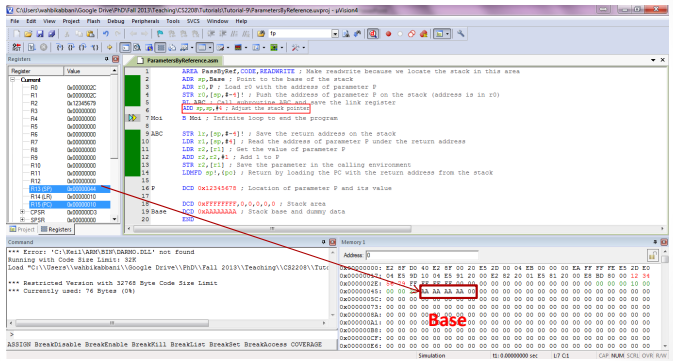
Passing By Reference

- In our last example, we forgot to do something important. What is it?
  - **Hint**: it involves the stack pointer.
- After the subroutine, **sp** was NOT pointing to the base.
- We should adjust the stack pointer. How?
  - ADD sp, sp, #4



39

Passing By Reference



40

Outline

- **Passing Parameters**
- **Value vs. Reference**
- **Passing By Value (revision)**
- **Passing By Reference**
- **Stack Frame**
- Conclusion

41

Stack Frame

- a subroutine sometimes needs local workspace for its temporary variables
- Each time the subroutine is called, a new workspace must be assigned to it
- The stack provides a convenient mechanism for implementing the dynamic allocation of workspace
- This storage allocation is dynamic because it is allocated to variables when they are created and then de-allocated when the variables are no longer required

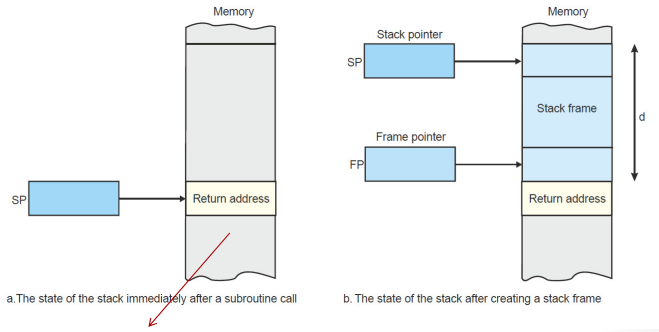
42

Stack Frame

- The tools to implement a local workspace are:
  - Stack Frame (SF)
    - a region of temporary storage at the top of the current stack
  - Frame Pointer (FP)
    - an address register
    - points to the bottom of the stack frame

43

Stack Frame



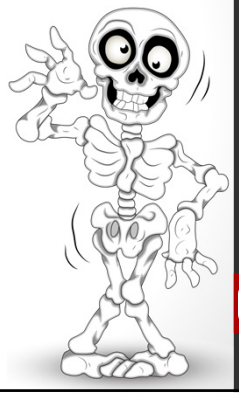
Question: What do we have here?

Answer: Parameters (if any) passed by the caller

44

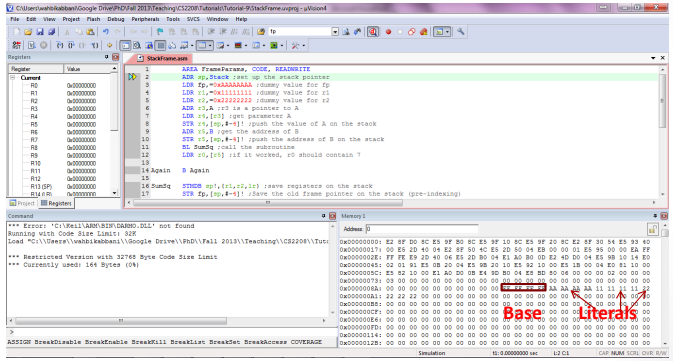
Stack Frame

- AnySub
  - STR fp,**[sp,#-4]!** Save the old frame pointer on the stack
  - MOV **fp**,sp Make FP point to the base of the stack frame
  - SUB **sp**,sp,#16 Make SP point to the top of the stack frame. **Note: 16 is just an example**
  - MOV **sp**,fp collapse the frame
  - LDR fp,**[sp],#4** Restore the old FP
  - MOV **pc**,lr Return



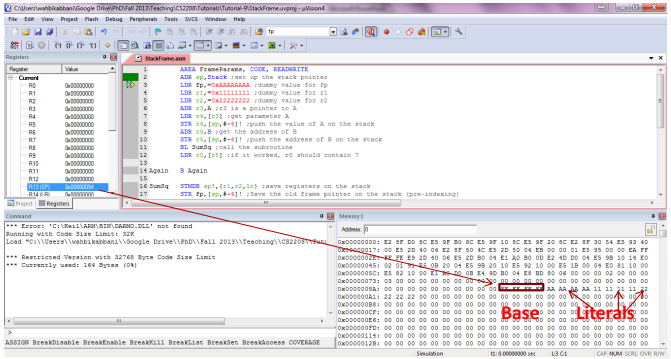
45

Stack Frame



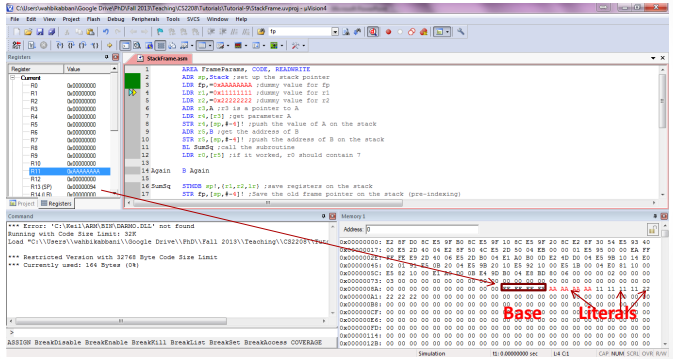
46

Stack Frame



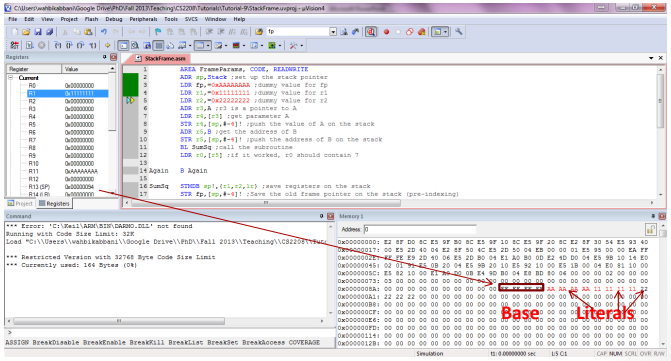
47

Stack Frame



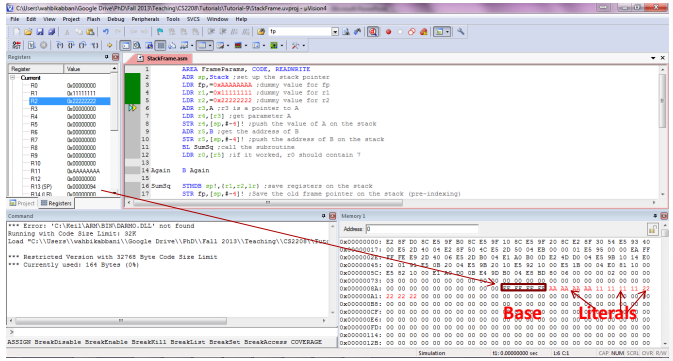
48

Stack Frame



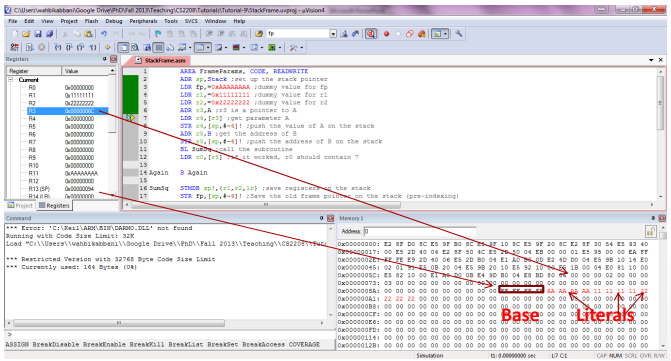
49

Stack Frame



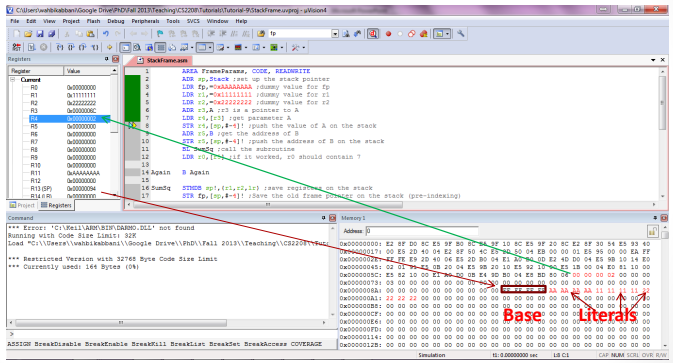
50

Stack Frame



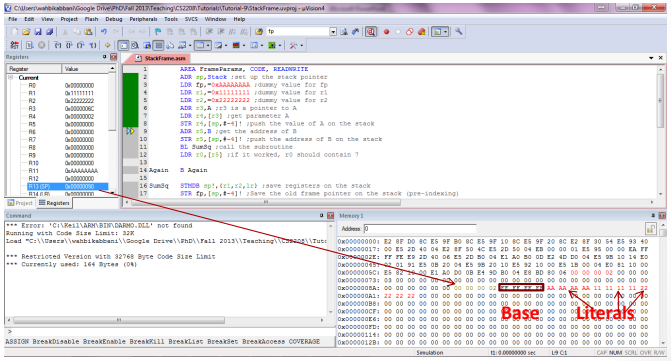
51

Stack Frame



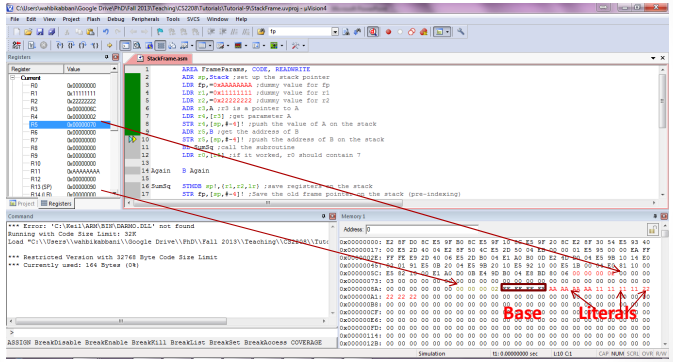
52

Stack Frame



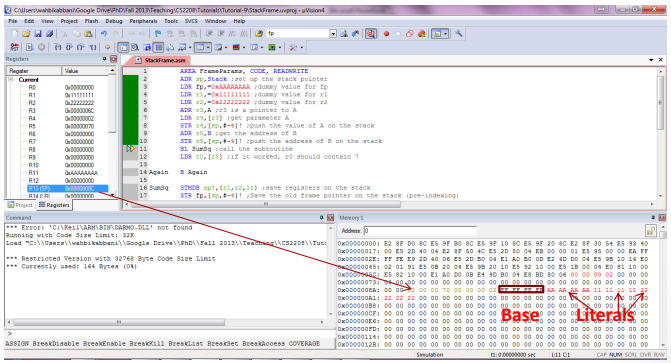
53

Stack Frame



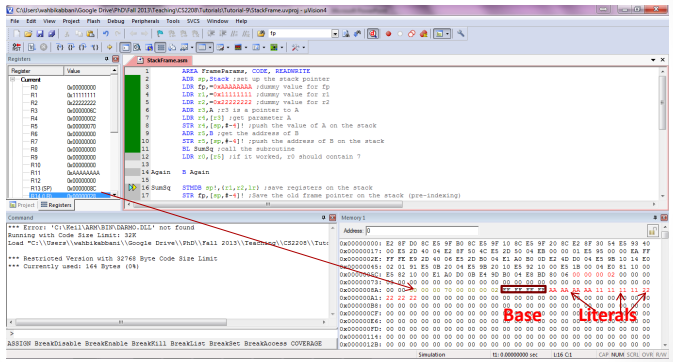
54

Stack Frame



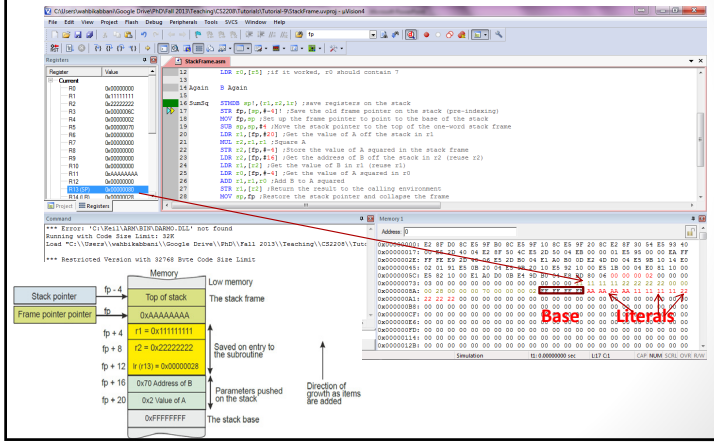
55

Stack Frame

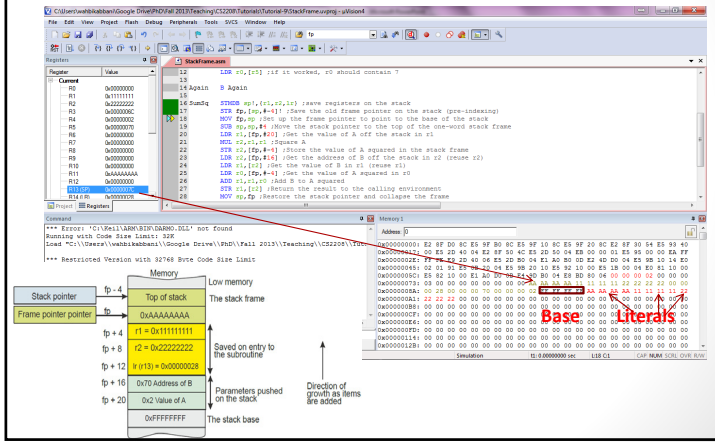


56

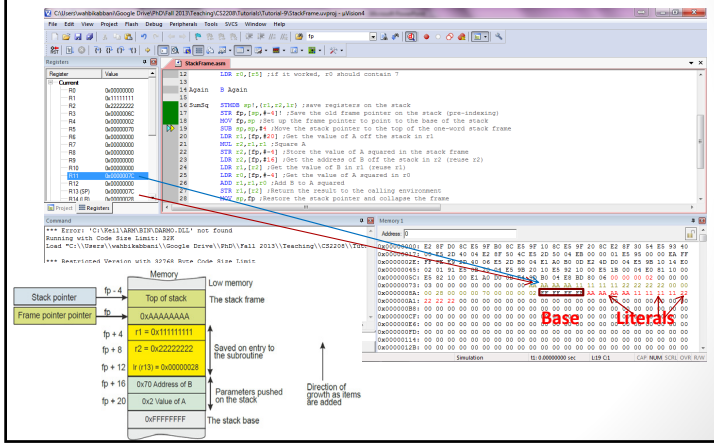
Stack Frame



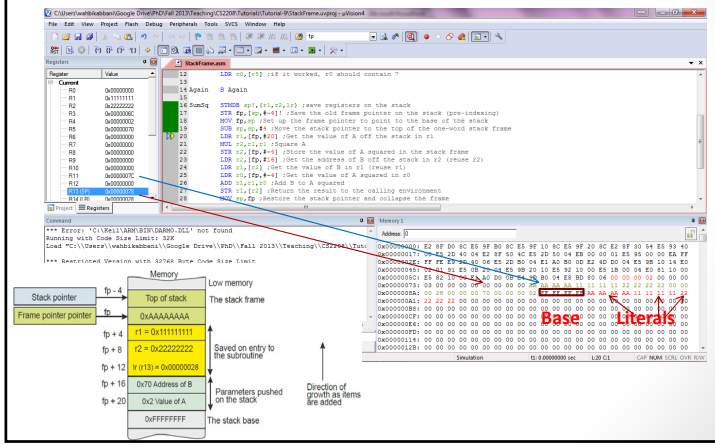
Stack Frame



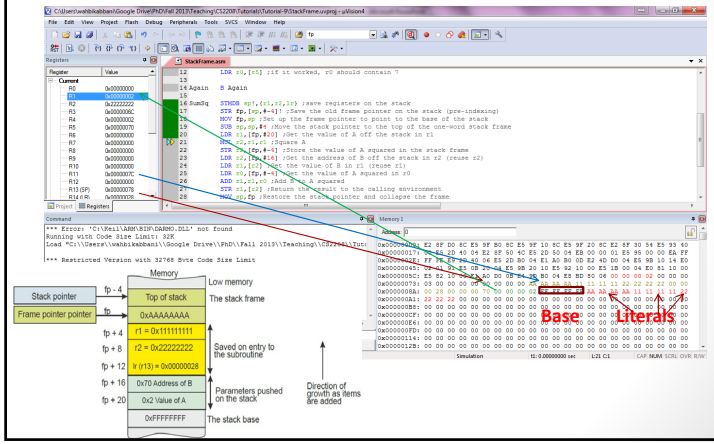
Stack Frame



Stack Frame

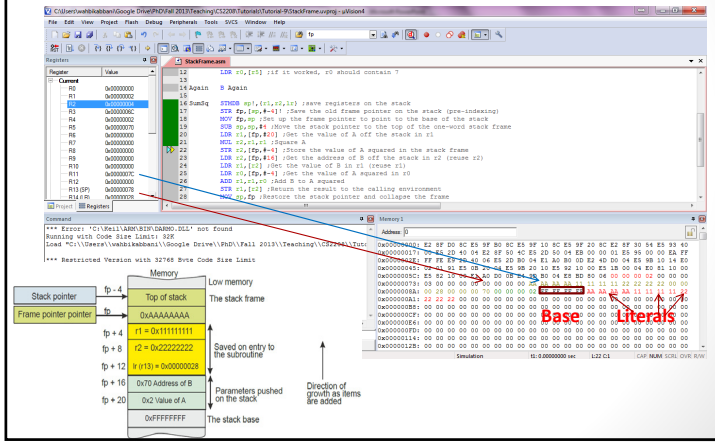


Stack Frame



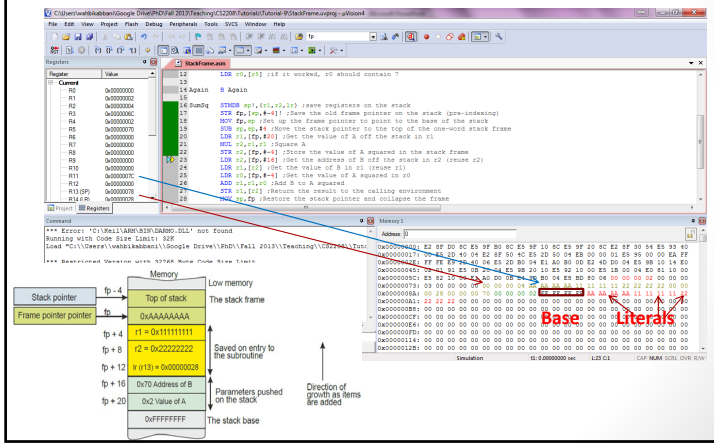
61

Stack Frame



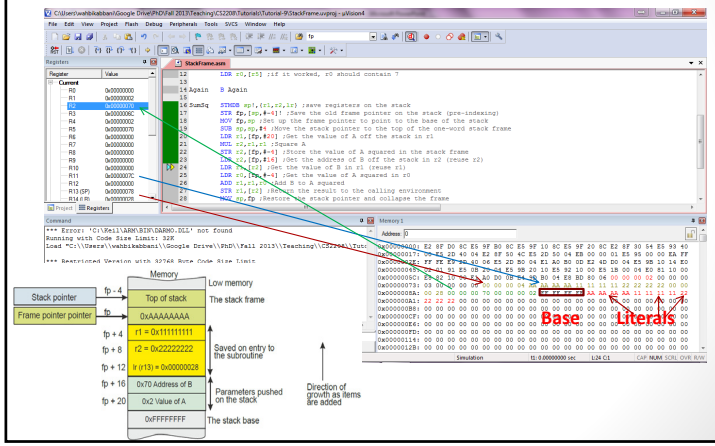
62

Stack Frame



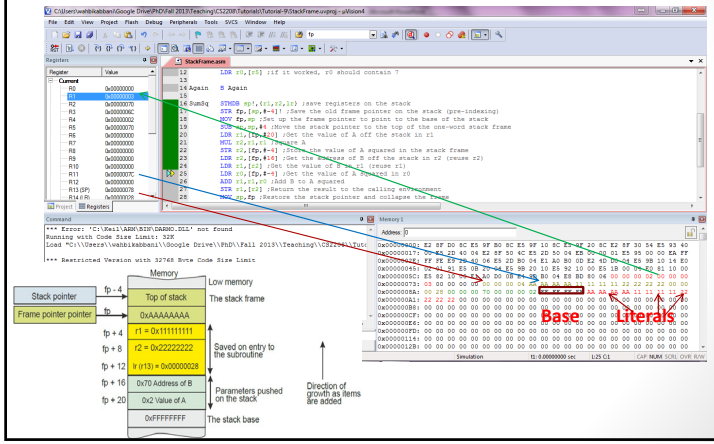
63

Stack Frame

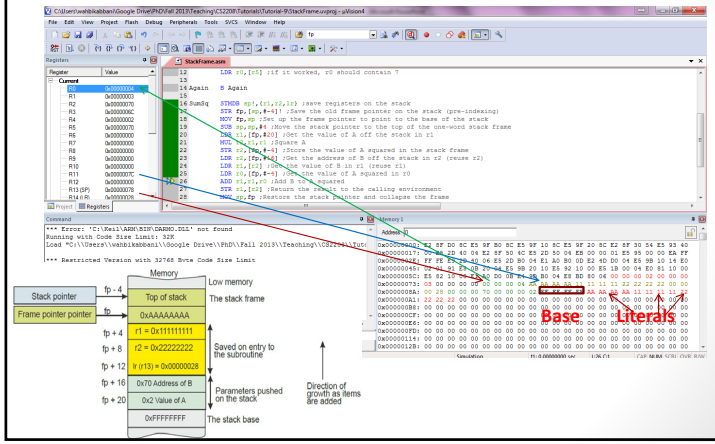


64

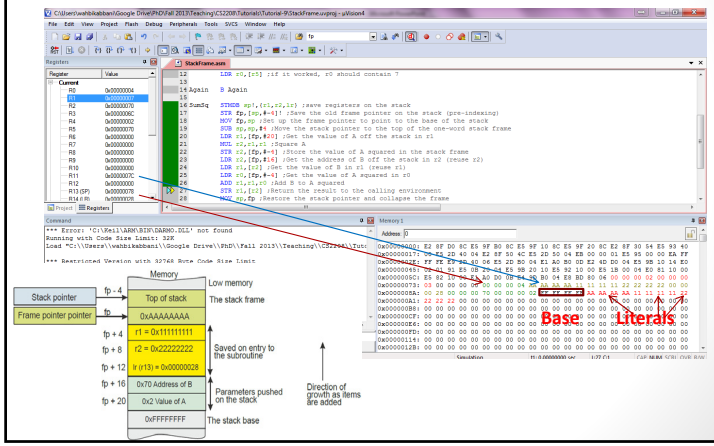
Stack Frame



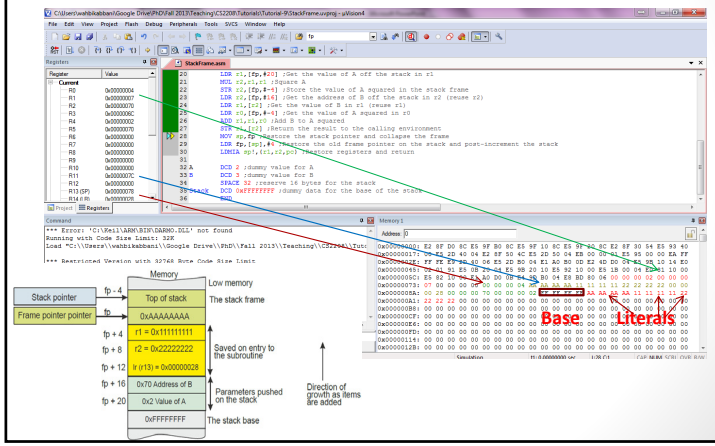
Stack Frame



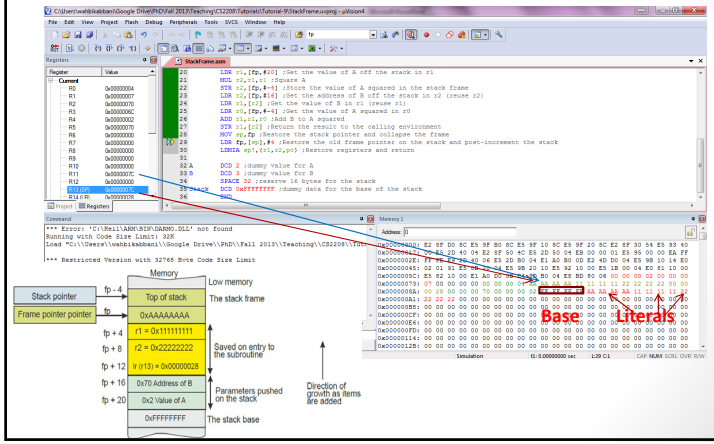
Stack Frame



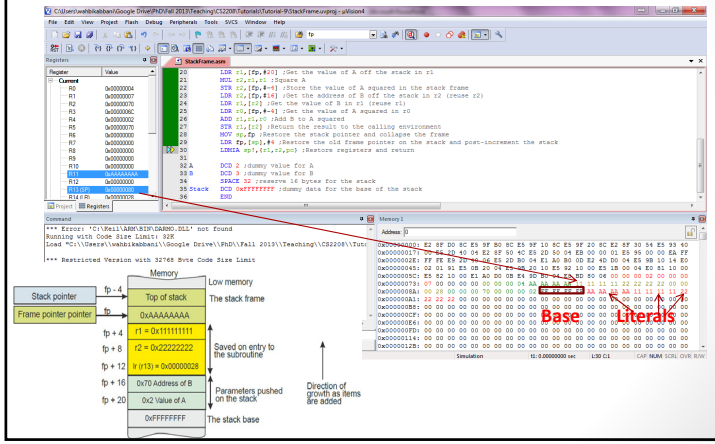
Stack Frame



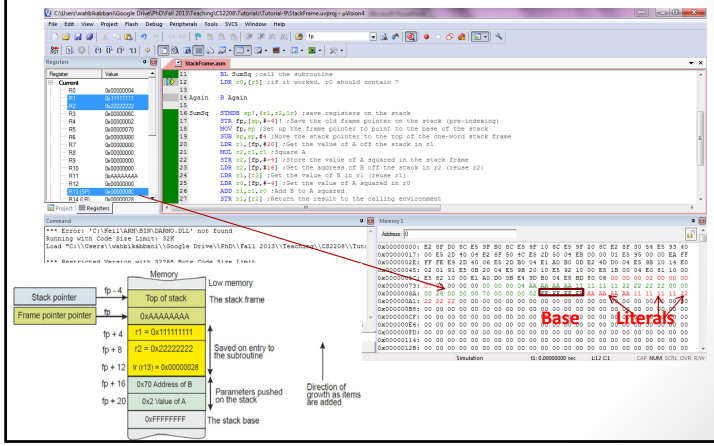
Stack Frame



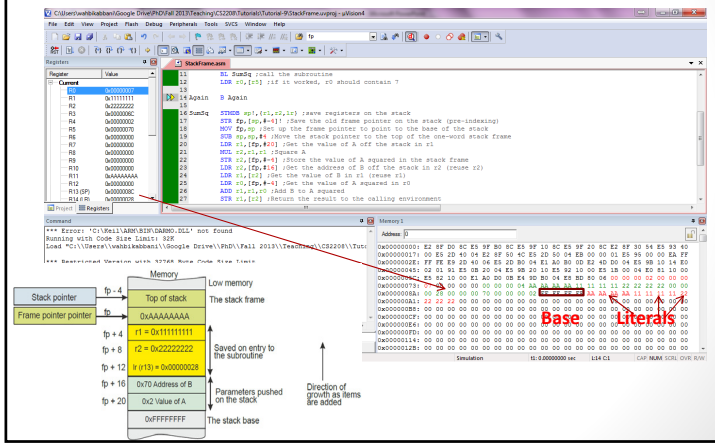
Stack Frame



Stack Frame



Stack Frame

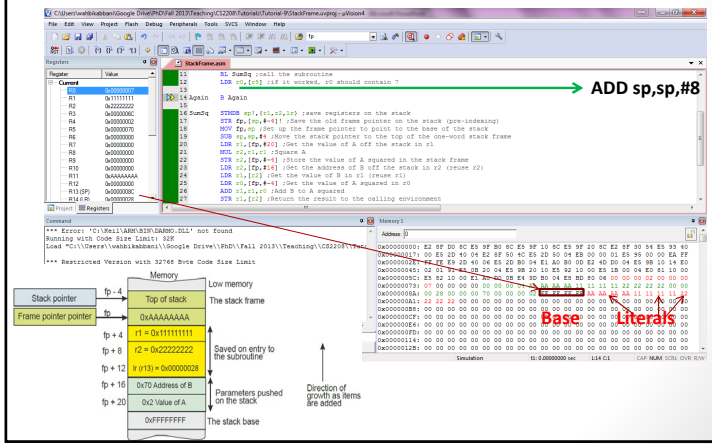


Stack Frame

- Again, what did we forget to do here?
  - Remove the parameters from the stack
  - How?
    - We have 2 parameters
    - ADD sp,sp,#8

73

Stack Frame



74

Outline

- **Passing Parameters**
- **Value vs. Reference**
- **Passing By Value (revision)**
- **Passing By Reference**
- **Stack Frame**
- **Conclusion**

75

Conclusion

- Passing By Value:
  - anything passed into a function call is unchanged in the caller's scope
- Passing By Reference:
  - A reference (memory address) to a variable is passed
  - The function can modify the parameter

76

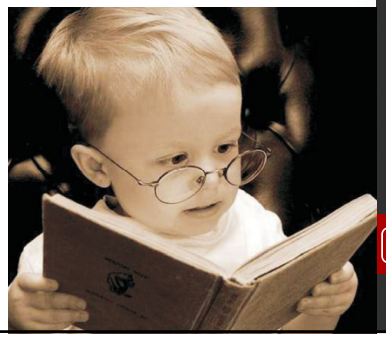
Conclusion

- a subroutine sometimes needs local workspace for its temporary variables
- Stack Frame (SF)
  - a region of temporary storage at the top of the current stack
- Frame Pointer (FP)
  - points to the bottom of the stack frame

77

The End

- Good Luck on your exams.
- Consultation Room : Middlesex College - 342



78