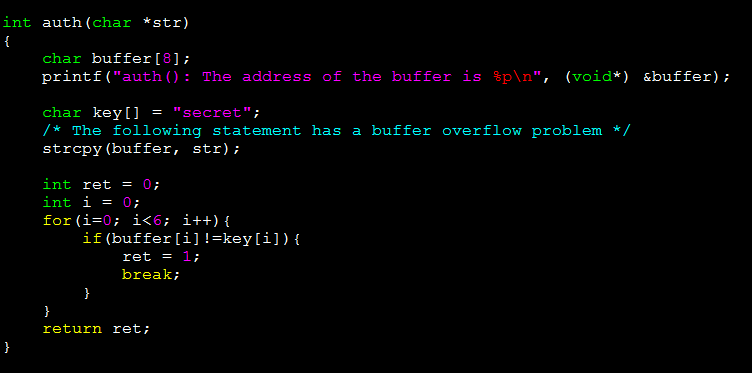
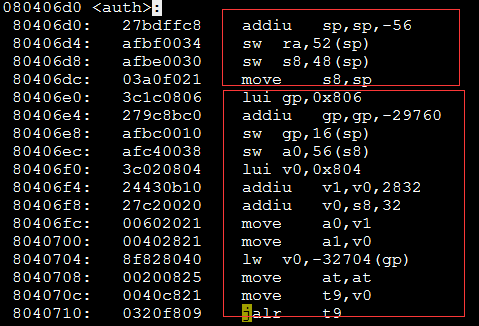


Before we do the three tasks in the Lab, we must know the content on the stack while program run into function `auth`. The compiled binary `vul\_program` print a very important message for us: the address of pointer buffer whose value is '0x7ff75a0'. Base on it, we can calculate some point address.

File `val\_program.dump` providers more detail for us:

Let me contrast `c` program with assembly program step by step:





Code in [80406d0-80406dc] show 56byte memory will be allocated on stack for function `auth`.

'sw ra,52(sp); sw s8, 48(sp)' is so important for use to solve task2 and task3. 'ra' store the the 'Return Address' for function 'auth'. Now, we can see the place on the stack. 's8' is the 'stack pointer' of the caller.

Code in [80406e0-8040710] corresponding to `printf` in `c` program. a0 save the first arguments, i.e. the const string "auth(): The address of the buffer is %p\n" which stored on static data segment. Attension please, "addiu v0,s8,32; move a1,v0" show what value is stored into a1.

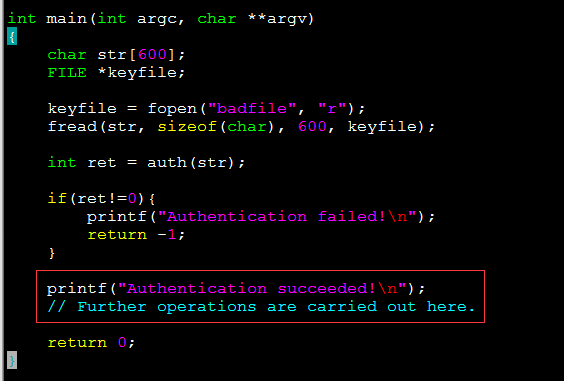
Because a1 is the second argument in printf, i.e. the address of pointer buffer. So we known,the buffer is placed on the stack[32(sp)]. So key should be placed on the stack[40(sp)].



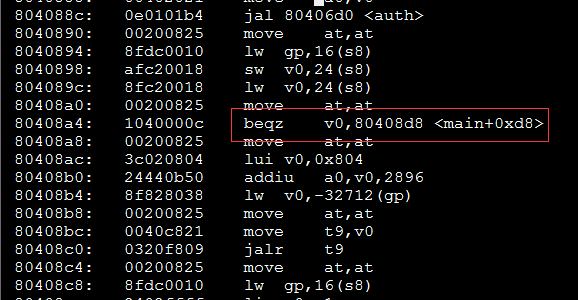
The graph above show the stack of 'main' and 'auth' when strcpy called. In auth stack, s8 should store the sp in main, so the value of it should be '0x7fff75b8'. str pointer to the buffer with size 600-bytes, we can get it's value easily in the graph: 0x7fff75b8 + (640-600-8) = 0x7fff75d8.

For task1, we just need make the first 6 letters in buffer is same as them in key, so, we put string "123456xx123456\x00" into the badfile. When strcpy called, the value of buffer is "123456xx" and the value of key if "123456\x00\x00". The value of key changed, so auth will success.

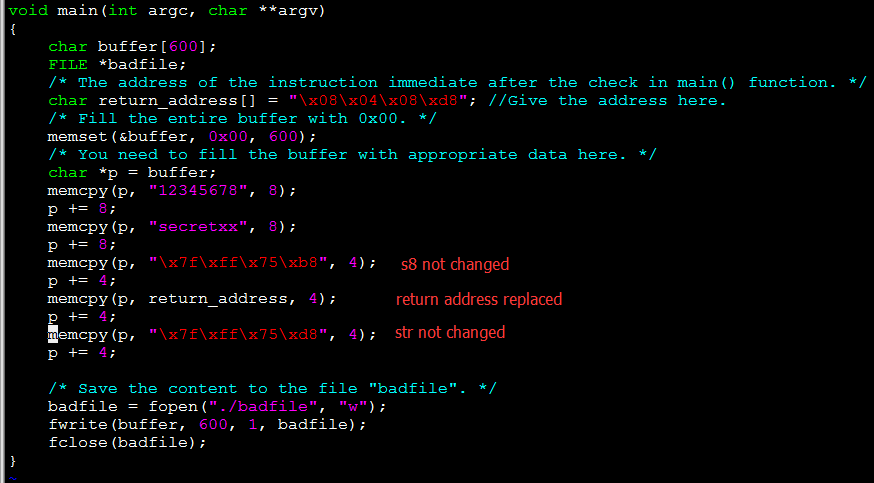
For task2, we should change the return address but not change s0 and str.



We wish no matter what value return from auth(str), the program will jump to the printf("Authentication succeeded!\n"). So we need get the address of the statement and replace the stack return address.



It is easy to get the address "0x080408d8", the code in task2.c show how to do it. Because the strcpy finished until get "0x00", and only the memory above on 'str' and be set '0x00', so we need get the original value of 'str' and set it again.



Task3 is similar to task2, we put the shell code just above 'str', so the address of the shell code is 0x7fff757c.

