CDA3101

Module 3: ARM Procedures (Part 1)

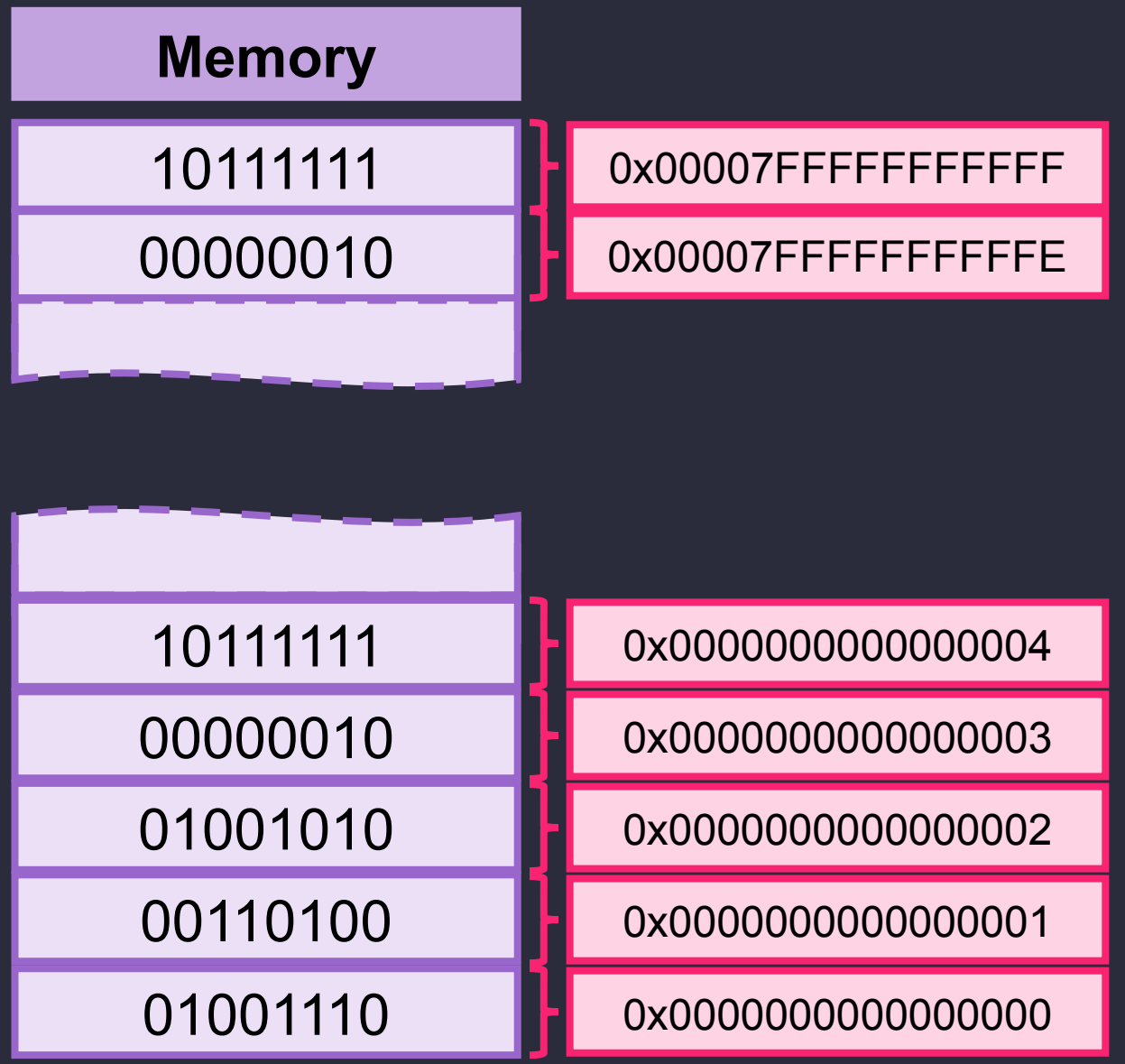
Welcome!

ARM Memory Model

Memory is a linear array of 2^{47} bytes.

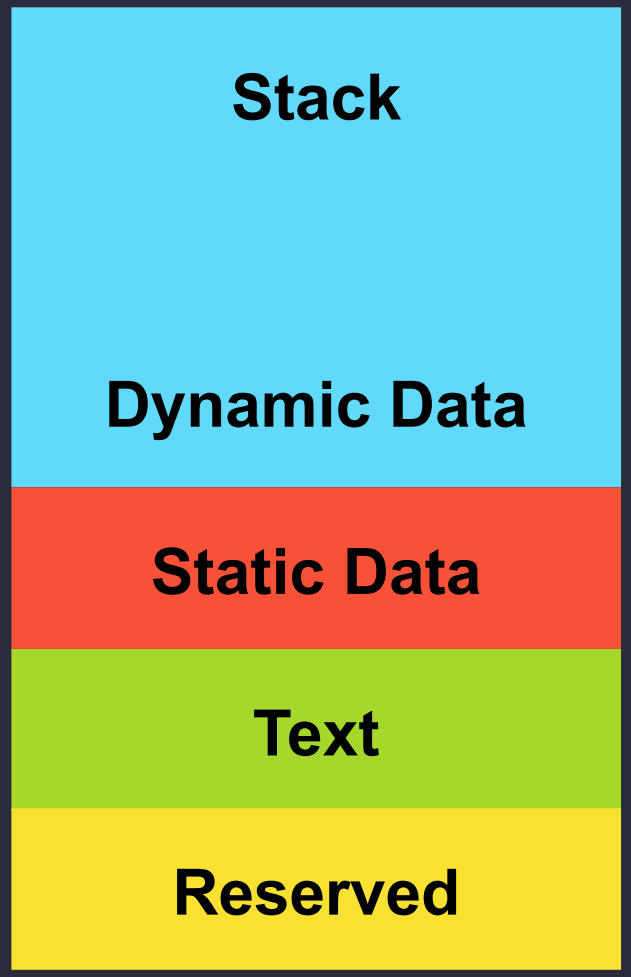
Each byte has a 64-bit address and can store an 8-bit pattern.

ARM addresses are 64 bits long and range from 0 to 2^{47}.



ARM Memory Model

SP 0000 007f ffff fffc$_{hex}$

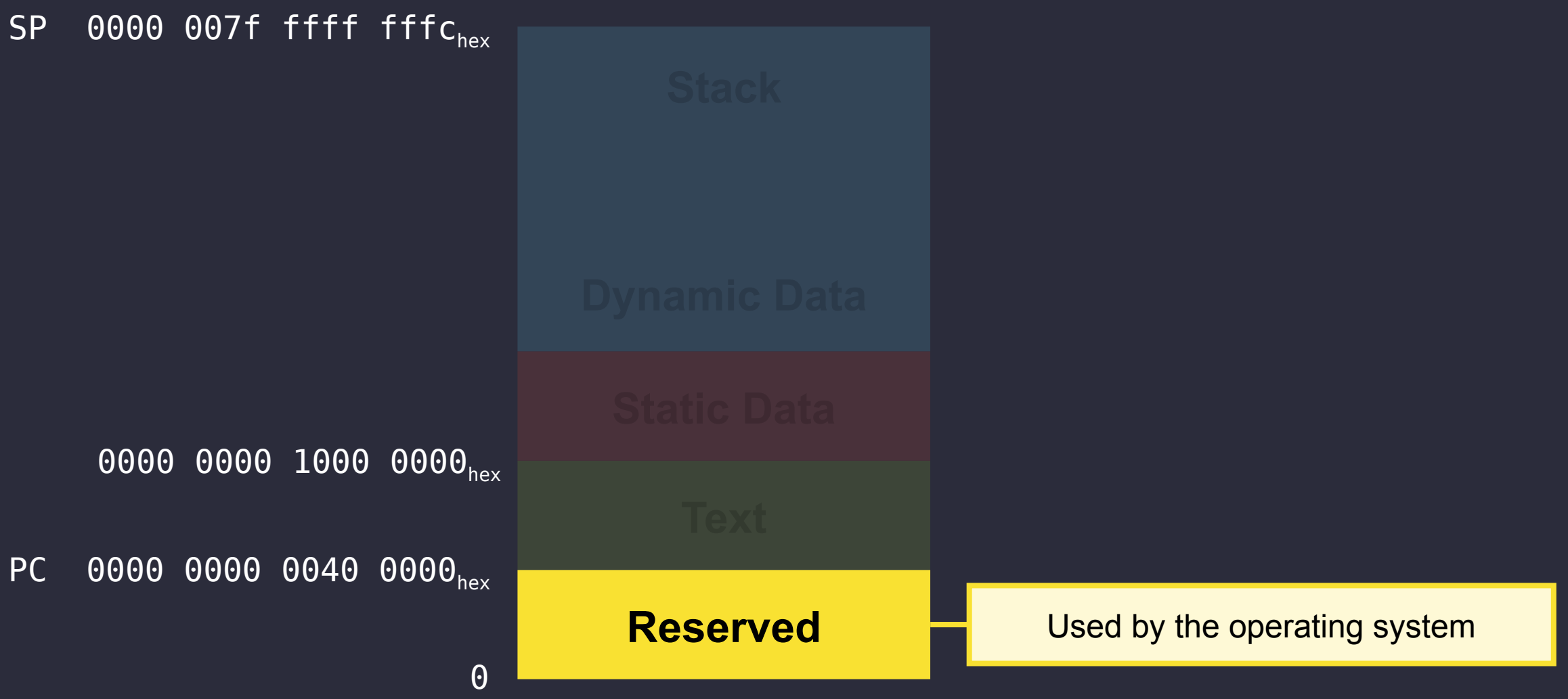


0000 0000 1000 0000$_{hex}$

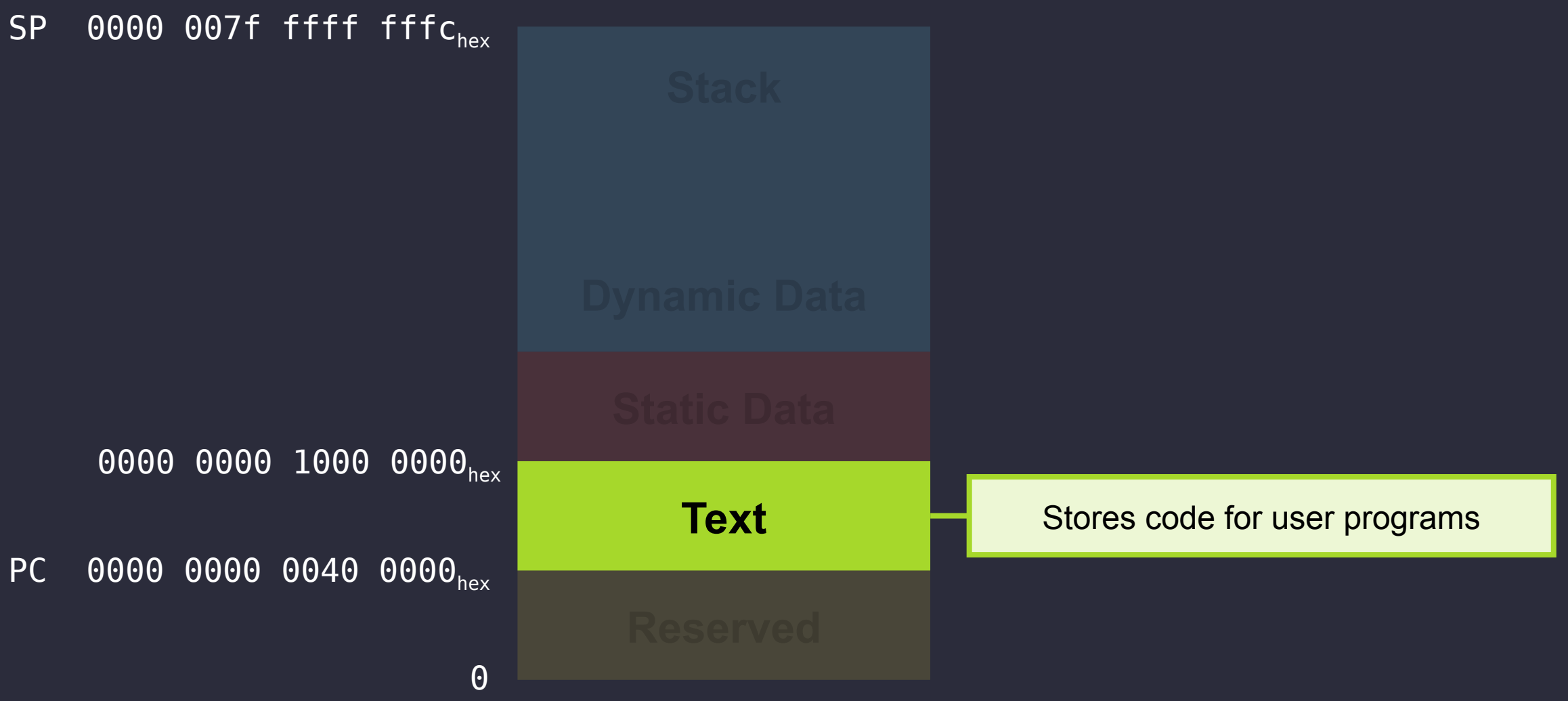
PC 0000 0000 0040 0000$_{hex}$

0

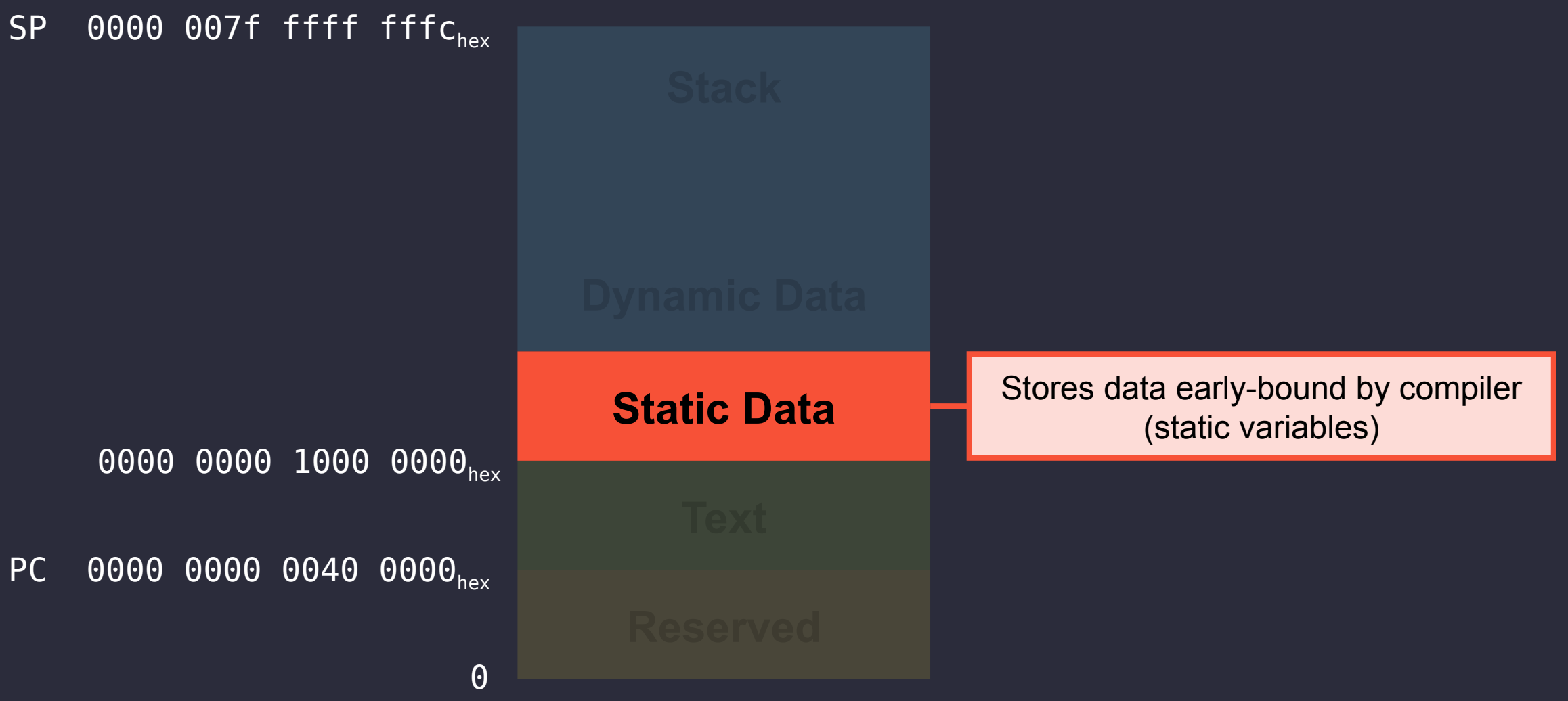
ARM Memory Model



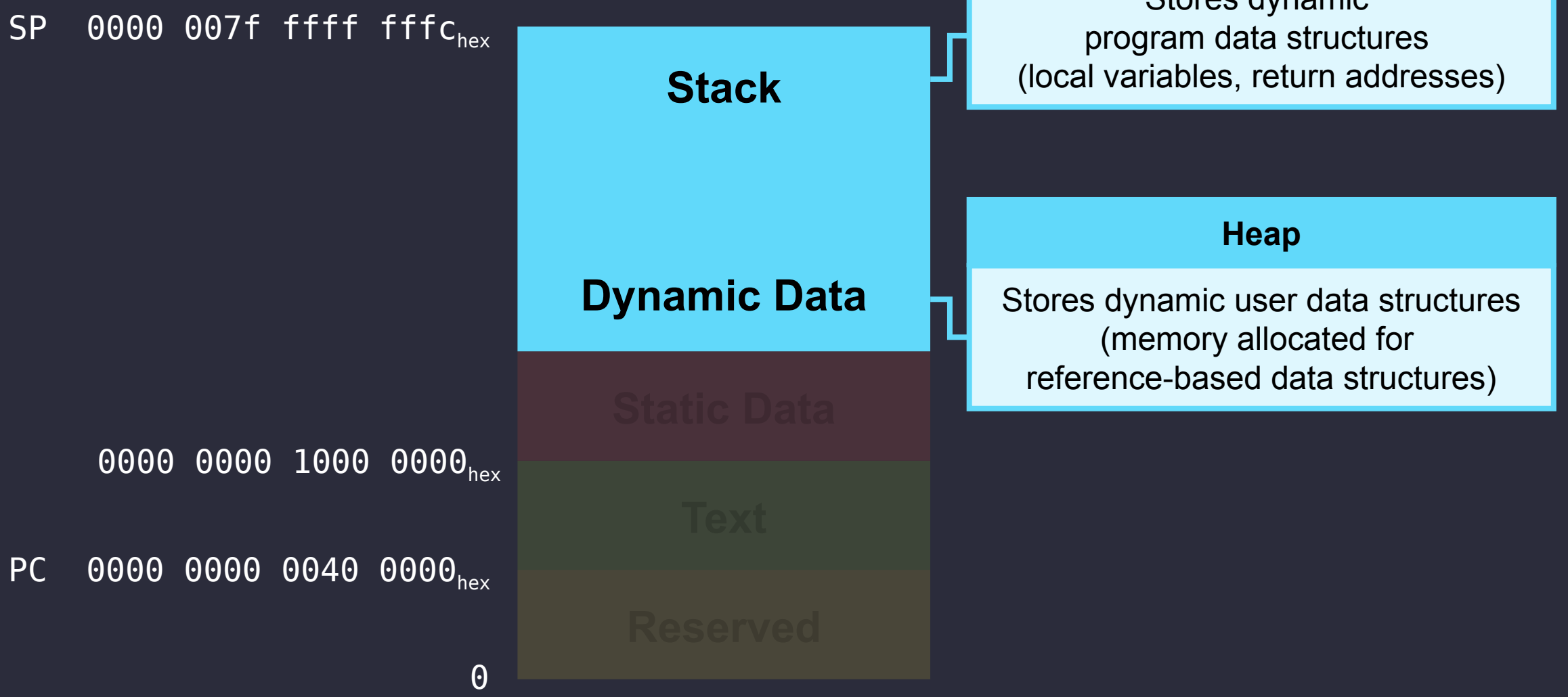
ARM Memory Model



ARM Memory Model



ARM Memory Model



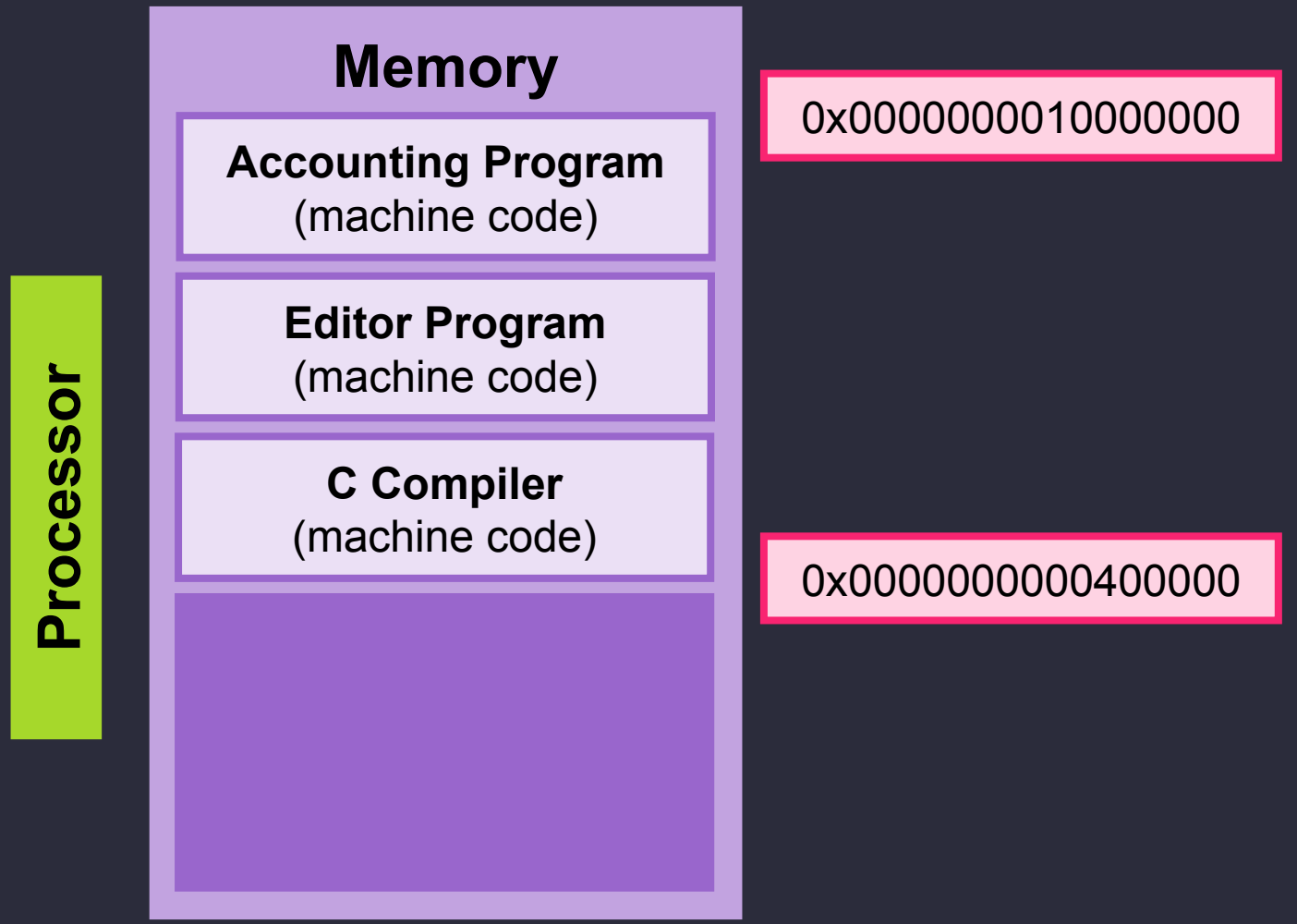
Stored Program Concept

The **Text** segment stores code for user programs.

Each machine code instruction is 32 bits (4 bytes).

The instruction's address will be the address of the first byte.

Subsequent instructions will have an address that is different by 4.



Stored Program Concept

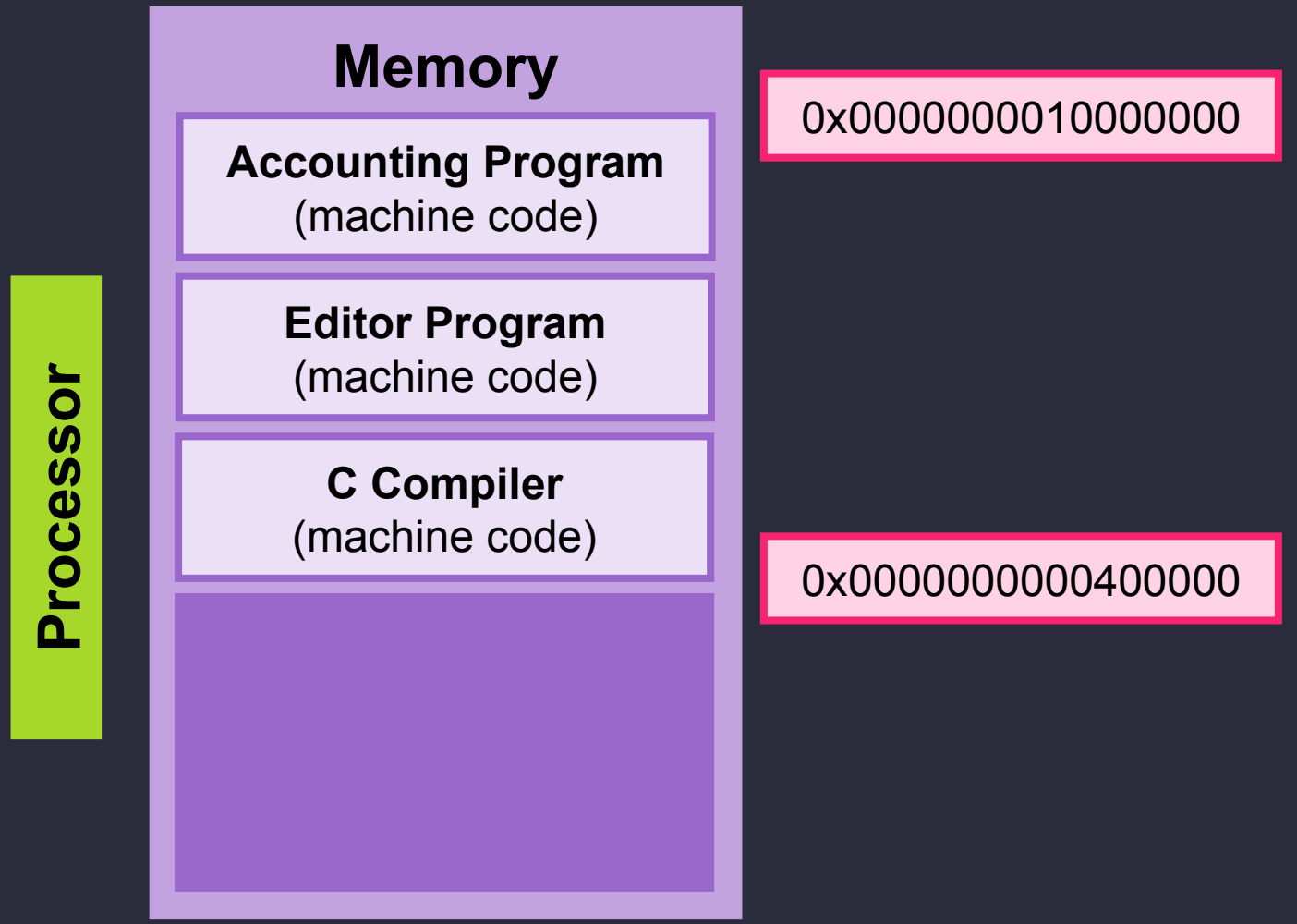
Instructions are **word aligned**. (word = 4 bytes)

Instructions must have an address divisible by 4.

Program Counter (PC) = address of the instruction being executed

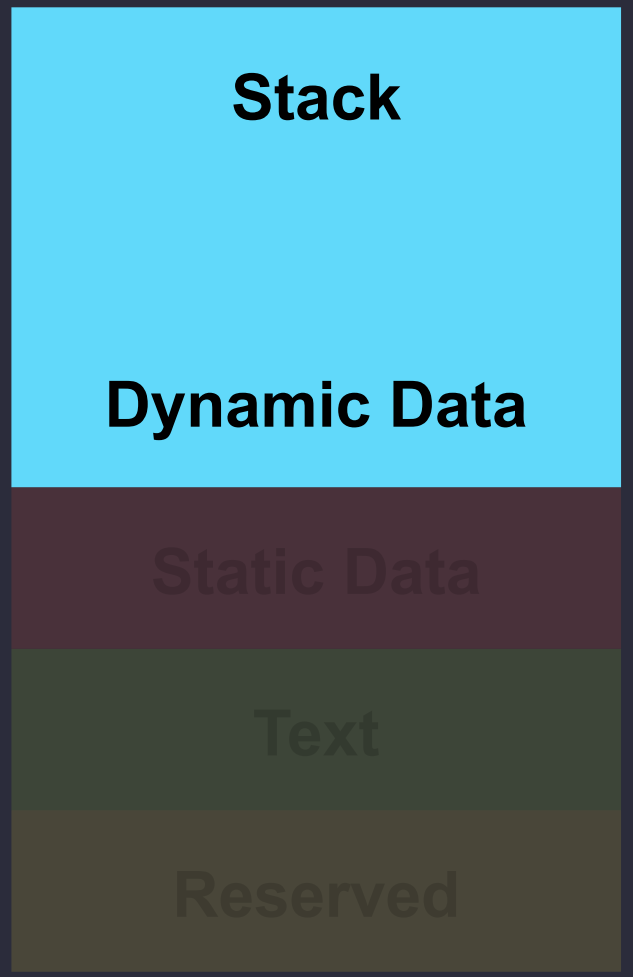
Add 4 if executing next instruction

Add offset if executing a branch



Stack Frame

SP 0000 007f ffff fffc$_{hex}$



0000 0000 1000 0000$_{hex}$

PC 0000 0000 0040 0000$_{hex}$

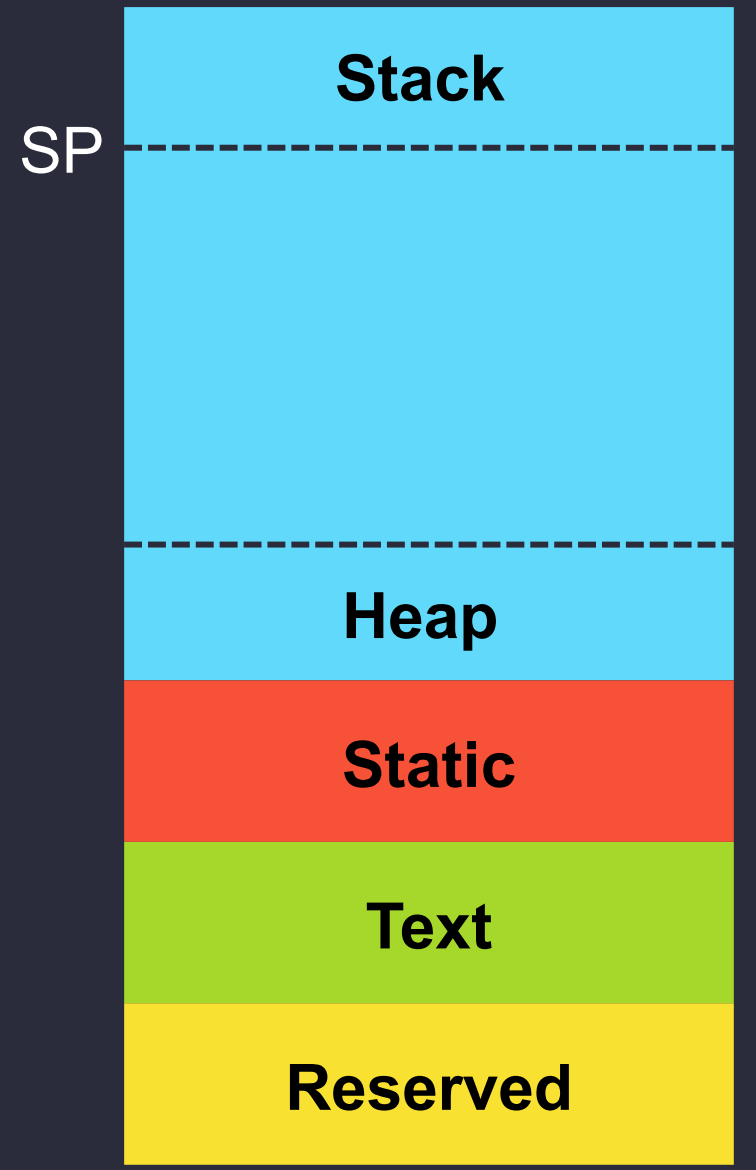
0

Stack Frame

A block of data that contains register contents and variables' values that you want to save when the procedure is called
(so the data is not lost)

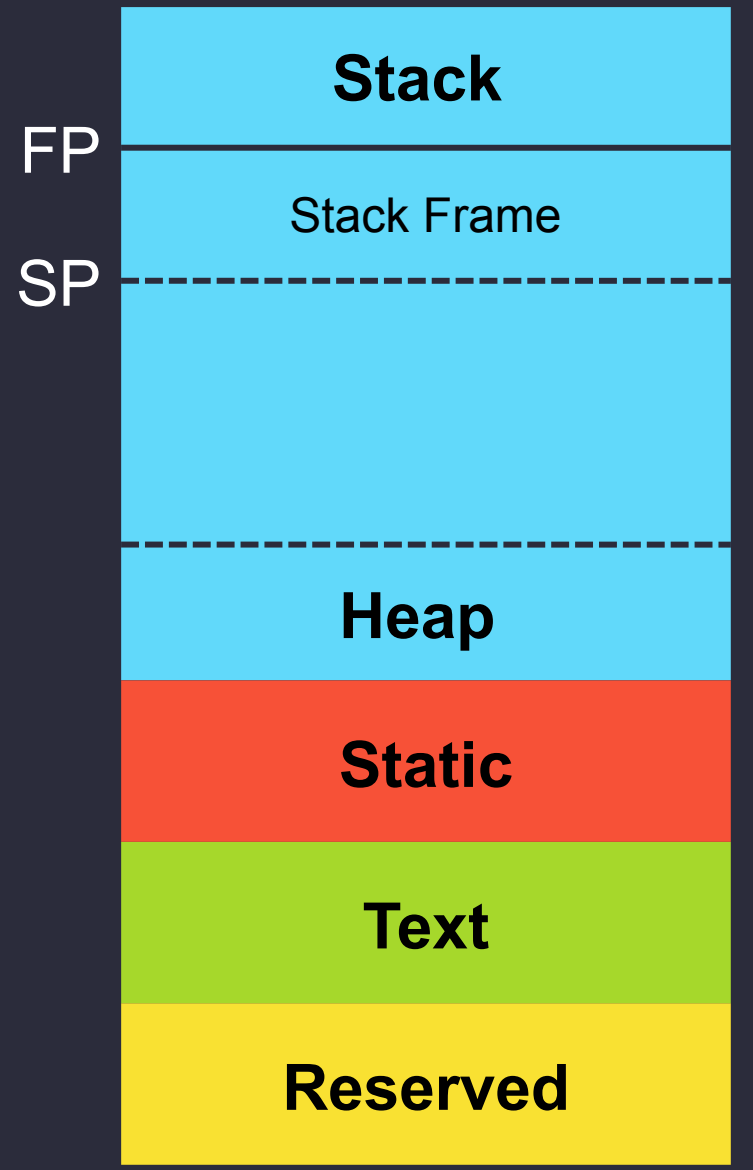
ARM Registers

Name	#	Use	Preserved across function call?
X0-X7	0-7	Arguments/results	No
X9-X15	9-15	Temporaries	No
X19-X27	19-27	Saved	Yes
SP	**28**	Stack pointer	Yes
FP	29	Frame pointer	Yes
X30	30	Return address	Yes
XZR	31	The constant zero	Yes



ARM Registers

Name	#	Use	Preserved across function call?
X0-X7	0-7	Arguments/results	No
X9-X15	9-15	Temporaries	No
X19-X27	19-27	Saved	Yes
SP	**28**	Stack pointer	Yes
FP	29	Frame pointer	Yes
X30	30	Return address	Yes
XZR	31	The constant zero	Yes



Translating Procedure Calls

```
int procedureA(int myArgument)
{
        int returnVariable, anotherVariable=2;

        returnVariable =
reB(anotherVariable);
        myArgument += returnVariable
        return myArgument;
}

int procedureB(int argument)
{
     int a;
    //code here
     return a;
}
```

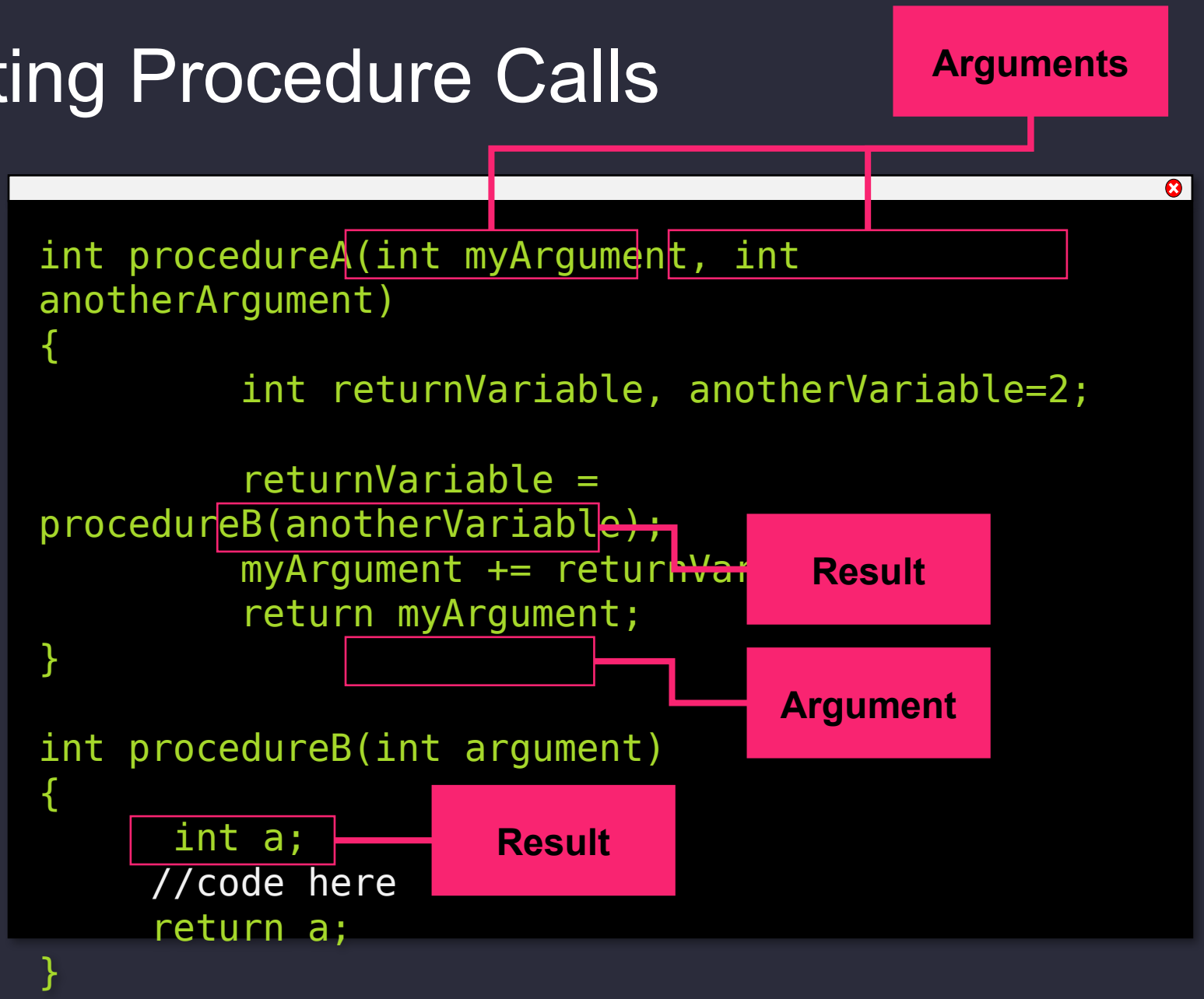
0x0000000010000000

0x0000000001000004

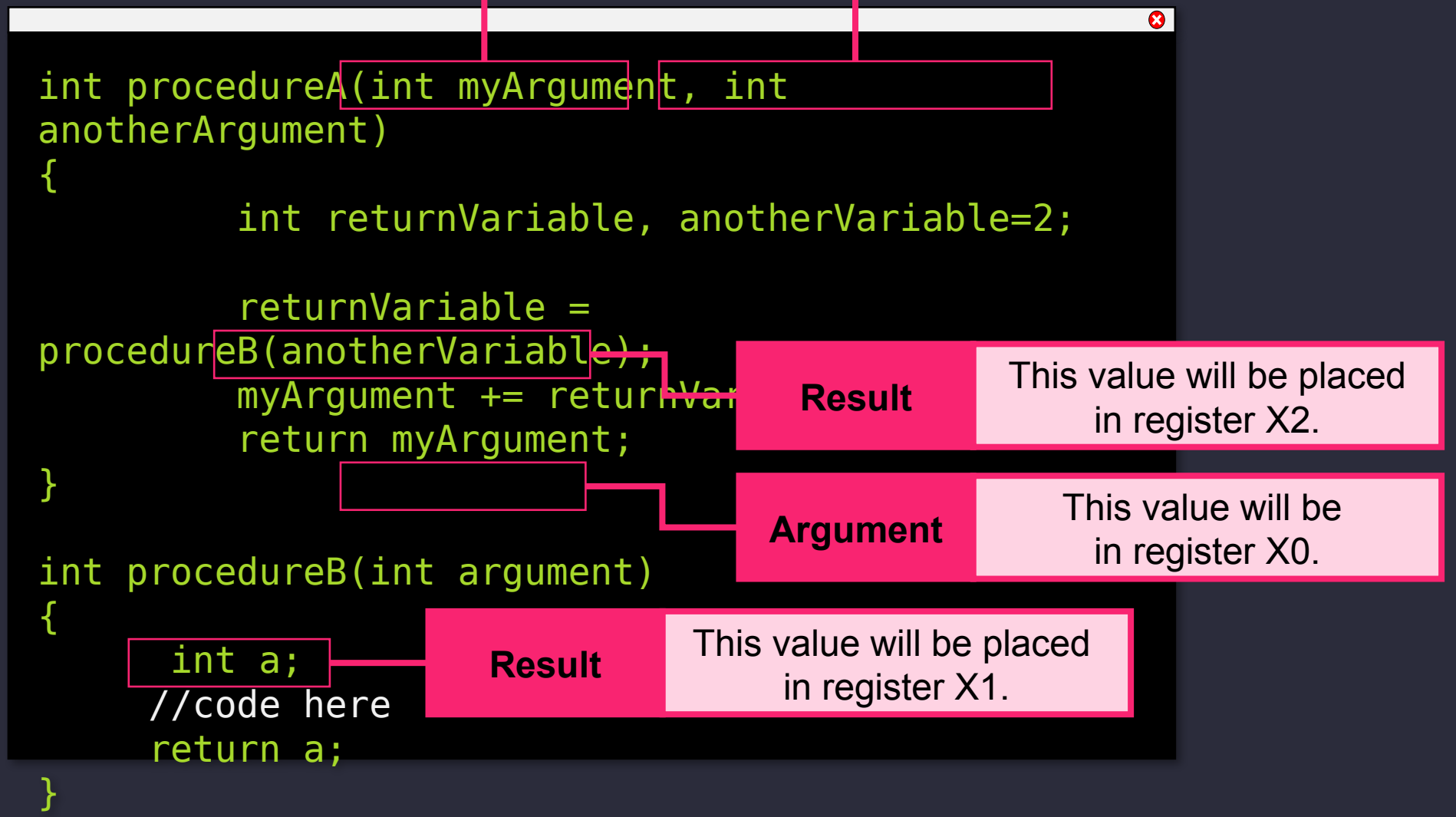
When calling myProcedure, the return address 0x0000000000100004 is put in register x30.

When this line executes, it returns to the address found in register x30.

Translating Procedure Calls



Translating Procedure Calls



Translating Procedure Calls

```
int procedureA(int myArgument, int
anotherArgument)
{
        int returnVariable, anotherVariable=2;

        returnVariable =
procedureB(anotherVariable);
        myArgument += returnVariabl
        return myArgument;
}

int procedureB(int argument)
{
     int a;
    //code here
    return a;
}
```

Caller

Callee

SP

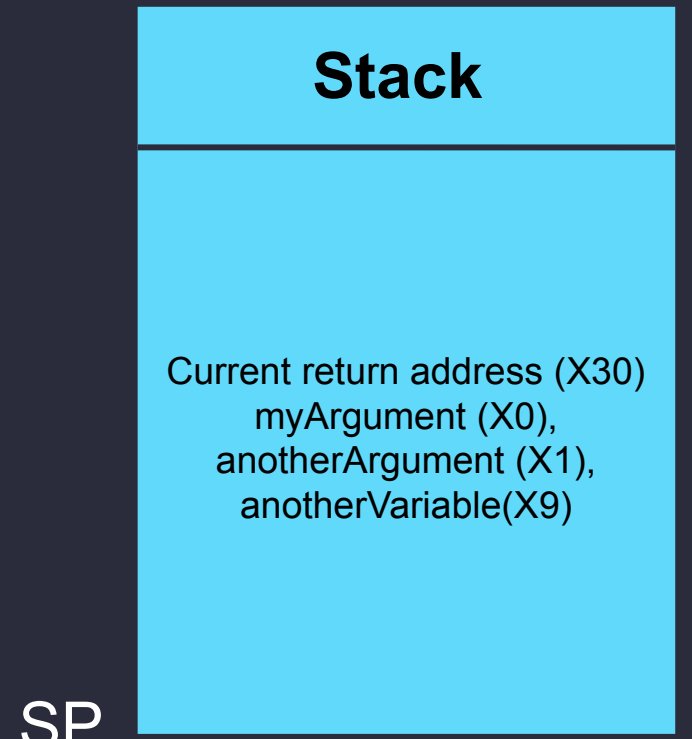
Stack

Translating Procedure Calls

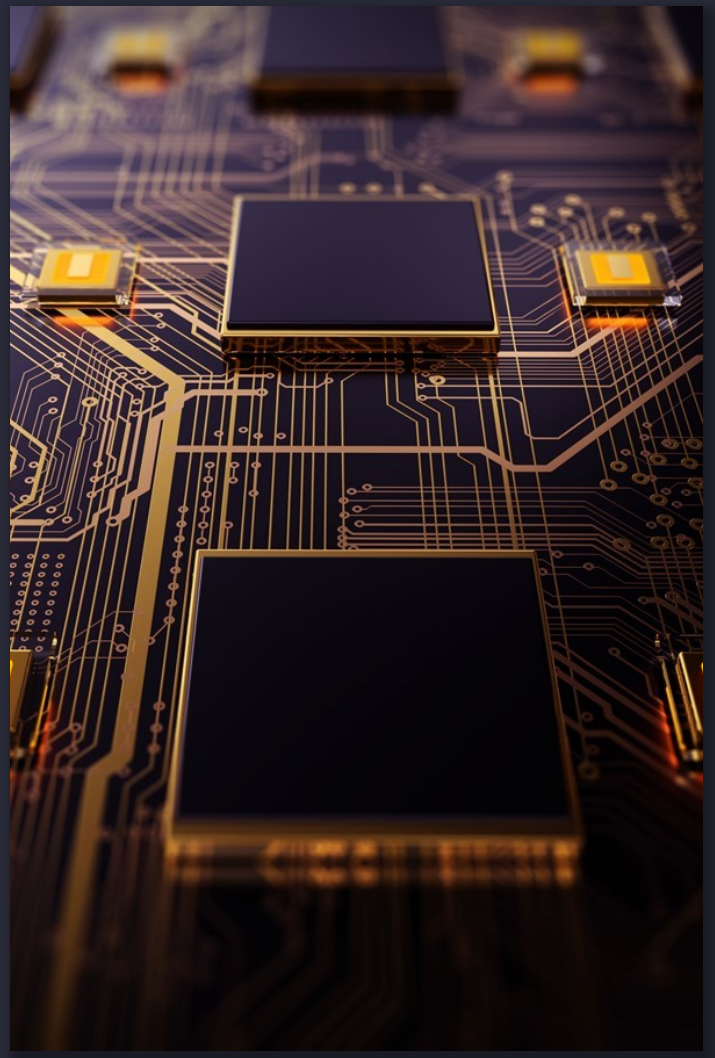
```
int procedureA(int myArgument, int
anotherArgument)
{
        int returnVariable, anotherVariable=2;

        returnVariable =
procedureB(anotherVariable);
        myArgument += returnVariabl
        return myArgument;
}

int procedureB(int argument)
{
     int a;
    //code here
    return a;
}
```

Caller

Callee



Wrap Up



Thank you for watching.

UF | UNIVERSITY of FLORIDA