ENGR-304-L

Software Lab 02

Agenda

1. Attendance
2. Overview
3. Lab Materials
4. Getting Started

Overview

Recap:

- Assembly code is assembled into machine code
- Assembly code and machine code are closely related

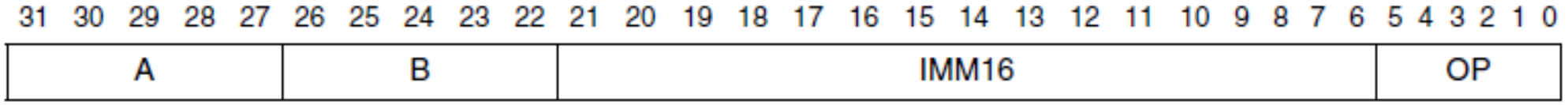
This Lab:

- Manually “assemble” a few lines of code to see the assembly process

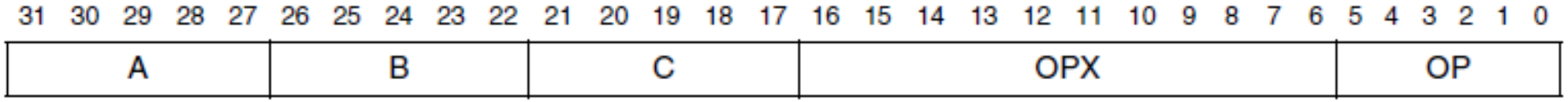
Overview

- NIOS Assembly has I, J, R Type Instructions
- Each type has a specific bitfield format for interpreting 32-bit data
- All types share the unique op-code field to identify which instruction

I Type



R Type



Overview

- Example: “add r8, r9, r10”

R Type – opcode 0x3A, opx 0x31

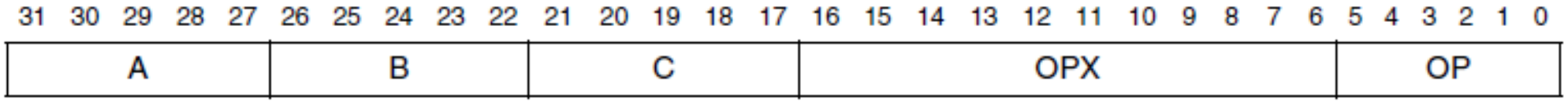
0x39	
0x3A	R-Type

OPX	Instruction
0x30	cmpltu
0x31	add

Overview

- Example: “add r8, r9, r10”

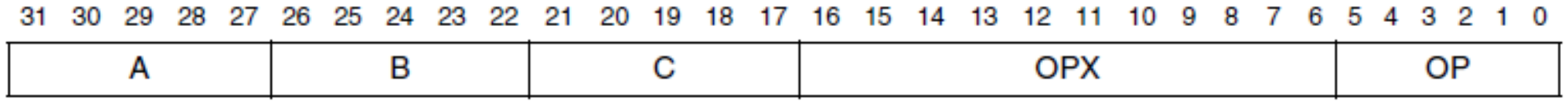
R Type – opcode 0x3A, opx 0x31



Overview

- Example: “add r8, r9, r10”

R Type – opcode 0x3A, opx 0x31



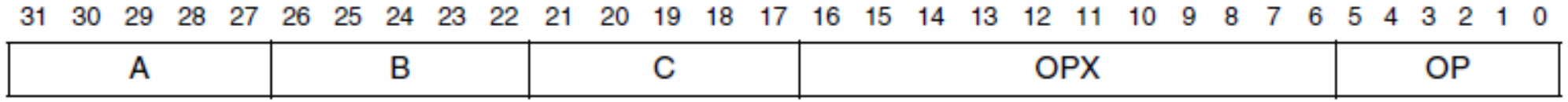
0b ? 0 0 0 0 0 1 1 0 0 0 1 1 1 1 0 1 0

Overview

Operation:	rC ← rA + rB
Assembler Syntax:	`add rC, rA, rB`

- Example: “add r8, r9, r10”

R Type – opcode 0x3A, opx 0x31



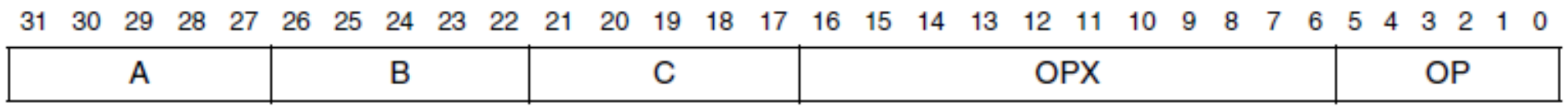
0b ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? 0 0 0 0 0 1 1 0 0 0 1 1 1 1 0 1 0

Overview

Operation:	rC ← rA + rB
Assembler Syntax:	`add rC, rA, rB`

- Example: “add r8, r9, r10”

R Type – opcode 0x3A, opx 0x31

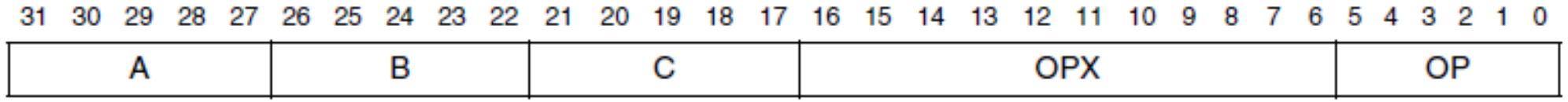


0b 0 1 0 0 1 0 1 0 1 0 0 1 0 0 0 0 0 1 1 0 0 0 1 1 1 1 0 1 0

Overview

- Example: “add r8, r9, r10”

R Type – opcode 0x3A, opx 0x31



0b 0 1 0 0 1 0 1 0 1 0 0 1 0 0 0 0 0 0 0 1 1 0 0 0 1 1 1 1 0 1 0

0x 4A90 0C7A

Overview

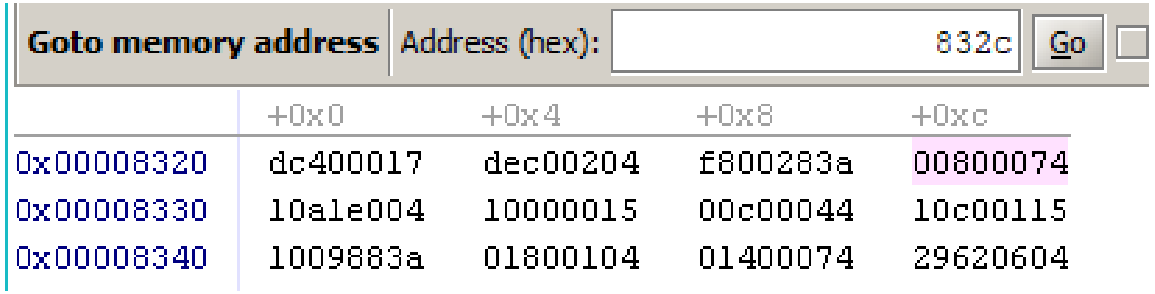
Memory Address Space

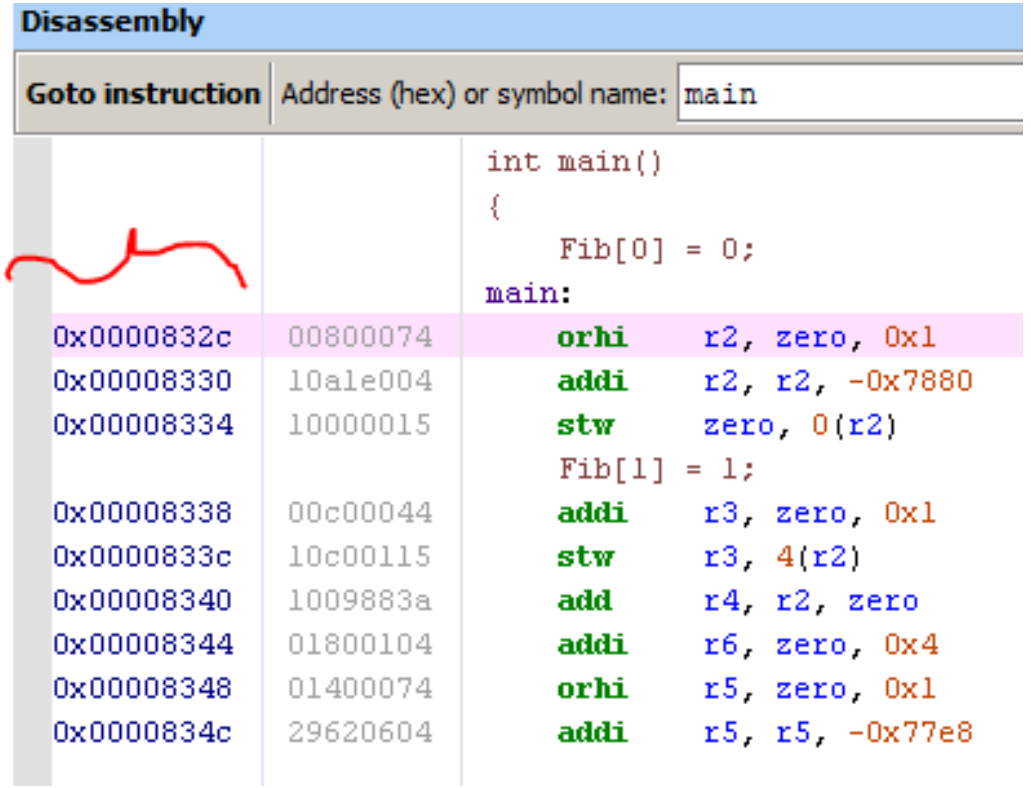
- Program instructions and data are stored in memory
- Memory is a larger and slower-to-access place to store data than registers

Overview

Memory Address Space

- We can view & change instruction machine code in memory
- We can view & change data in memory





Overview

Memory Address Space

- We can view & change instruction machine code in memory
- We can view & change data in memory

	+0x0	+0x4	+0x8	+0xc
0x00008760	0	0	0	0
0x00008770	0	0	33656	33656
0x00008780	0	1	1	2
0x00008790	3	5	8	13
0x000087a0	21	34	55	89
0x000087b0	144	233	377	610
0x000087c0	987	1597	2584	4181
0x000087d0	6765	10946	17711	28657
0x000087e0	46368	75025	121393	196418
0x000087f0	317811	514229	832040	1346269
0x00008800	2178309	3524578	5702887	9227465
0x00008810	14930352	24157817	39088169	63245986
0x00008820	0	0	0	0

	+0x0	+0x4	+0x8	+0xc
0x00008760	00000000	00000000	00000000	00000000
0x00008770	00000000	00000000	00008378	00008378
0x00008780	00000000	00000001	00000001	00000002
0x00008790	00000003	00000005	00000008	0000000d
0x000087a0	00000015	00000022	00000037	00000059
0x000087b0	00000090	000000e9	00000179	00000262
0x000087c0	000003db	0000063d	00000a18	00001055
0x000087d0	00001a6d	00002ac2	0000452f	00006ff1
0x000087e0	0000b520	00012511	0001da31	0002ff42
0x000087f0	0004d973	0007d8b5	000cb228	00148add
0x00008800	00213d05	0035c7e2	005704e7	008cccc9
0x00008810	00e3d1b0	01709e79	02547029	03c50ea2
0x00008820	00000000	00000000	00000000	00000000

Overview

Assembly Usage of Memory

- Declare data with initial values
- Declare uninitialized data

```
/*********************/
/*******...DATA....*****/
/*********************/
/* The ".data" directive identifies the section
.data

MyArray:
.word 52,377,136,2011,23,872,1003,1,97,5432,0

/* The ".end" assembler directive indicates the
   ...all following lines are discarded */
.end
```

```
/*********************/
/*******...DATA....*****/
/*********************/
/* The ".data" directive identifies the section of the pro
.data

Fib: /* uninitialized memory of size 40 4-byte values */
.skip 160

/* The ".end" assembler directive indicates the end of the
   ...all following lines are discarded */
.end
```

Overview

Assembly Usage of Memory

- Use stw to store a word (32bits) into memory at an address from a register
- Use ldw to load a word from memory to a register
- The gp register is intended to hold memory addresses, the ldw and stw instructions require the address to be in a register
- Note immediate value offsets are also available in ldw/stw, may be 0

```
··movia·gp,·Fib···/*·point·gp·to·start·of·Fib·memory·*/
```

Overview

Optimization Considerations

- Factors: code footprint (instructions in program) vs

  code performance (instructions executed)

- Compilers can optimize code to run faster with a smaller footprint, often useful but sometimes risky
- For programs that are highly resource constrained, a careful & experienced assembly programmer can “beat” the compiler

Lab Materials

- NIOS CPU Summary
- NIOS CPU Reference

- Notepad++ & Assembly Profile
- Altera DE2 Boards & NIOS CPU Image

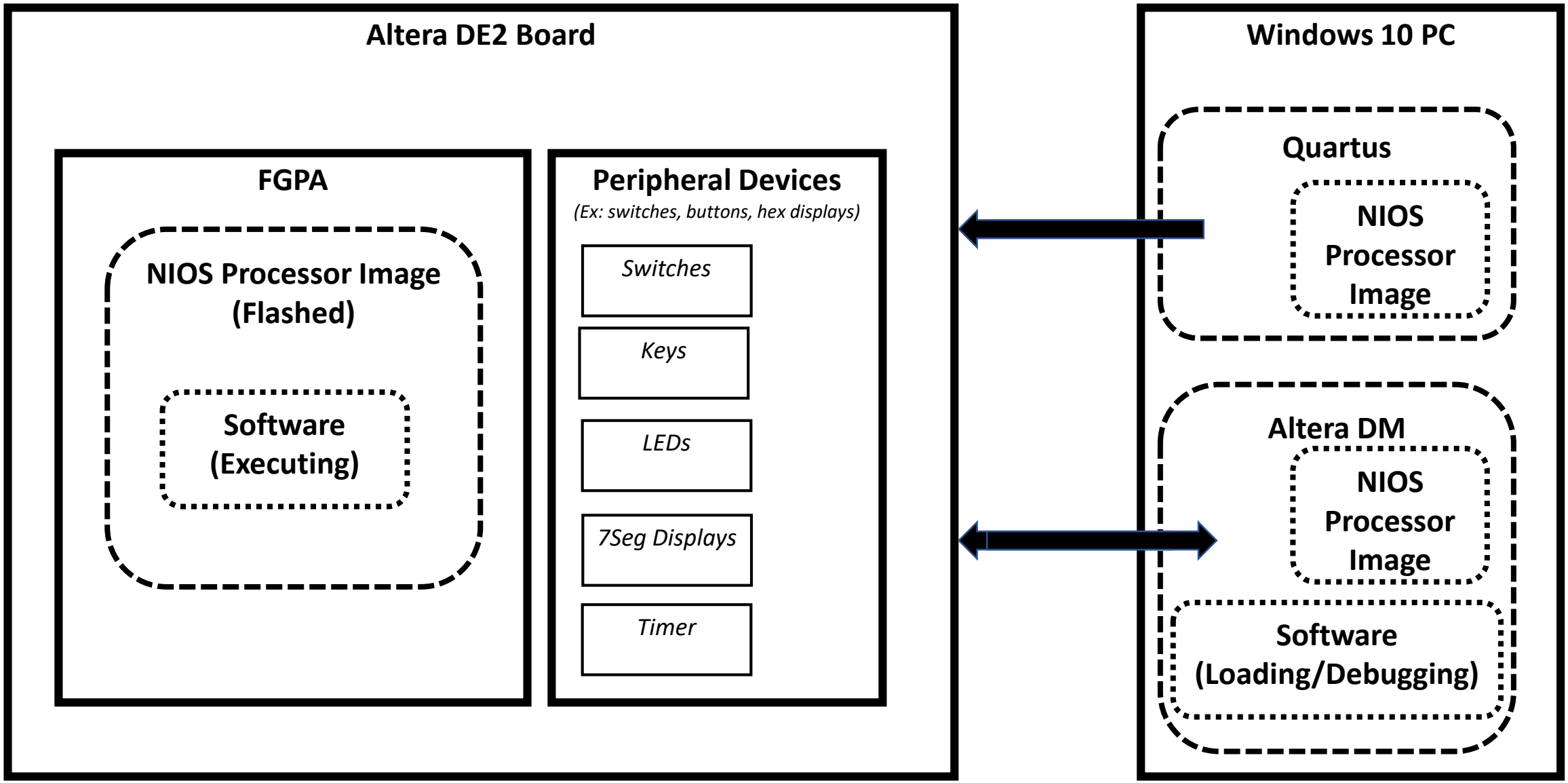
- Quartus
- Altera Debug Monitor

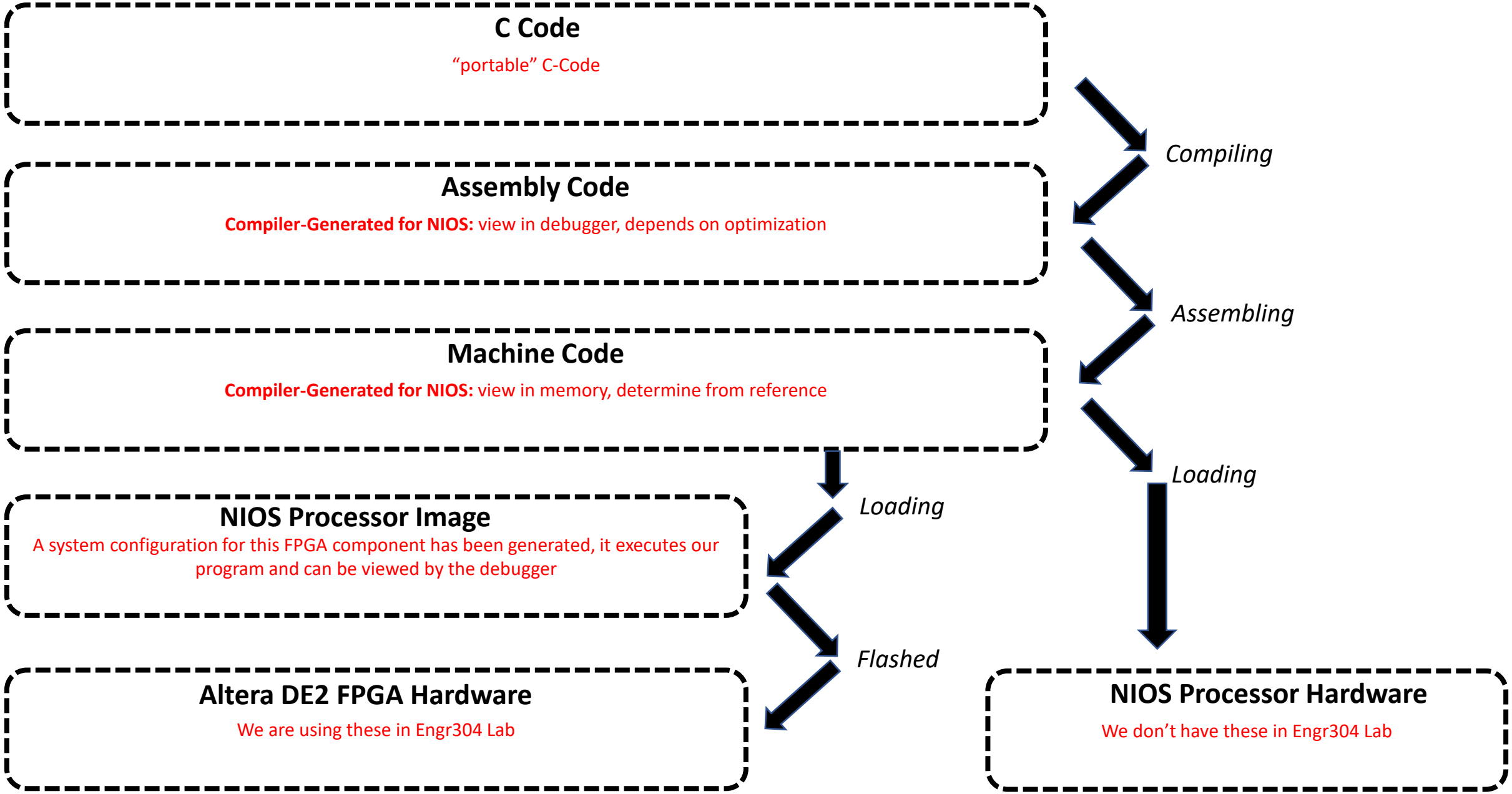
Getting Started

1. Setup project directory for SW02 on the **H: drive**
2. Download SW02 files from Moodle or S: drive
3. Also locate the reference files from SW01
4. Start following directions in the lab assignment document

Reference Diagrams

- The remaining slides explain certain file types and compilation processes that will apply throughout the semester


Altera DE2 Board
FGPA
NIOS Processor Image (Flashed)
Software (Executing)
Peripheral Devices
(Ex: switches, buttons, hex displays)
Switches
Keys
LEDs
7Seg Displays
Timer
Windows 10 PC
Quartus
NIOS Processor Image
Altera DM
NIOS Processor Image
Software (Loading/Debugging)


C Code
"portable" C-Code
Compiling
Assembly Code
Compiler-Generated for NIOS: view in debugger, depends on optimization
Assembling
Machine Code
Compiler-Generated for NIOS: view in memory, determine from reference
Loading
Loading
NIOS Processor Image
A system configuration for this FPGA component has been generated, it executes our program and can be viewed by the debugger
Flashed
Altera DE2 FPGA Hardware
We are using these in Engr304 Lab
NIOS Processor Hardware
We don't have these in Engr304 Lab

C Code

C Code is fairly platform-agnostic and can be run on a variety of difference processors, each with their own assembly instruction sets. C Code is compiled into assembly code which is beneficial because the combination of C Code and a compiler has better maintainability, portability, readability, and shorter development times than raw assembly code.

Compiling

Assembly Code

Assembly code is fairly platform-specific and may be shared by families of processors. Assembly code is almost at the level of machine code, but is more readable and maintainable than raw 0's and 1's. It is assembled into machine code.

Assembling

Machine Code

Machine code is the raw 0's and 1's that can be interpreted by the processor in order to execute programs. It is rarely used or modified directly since changes to source code occur at the C Code or Assembly Code levels instead. However, it may be captured as an artifact of the build process for a particular program.

Loading

Loading

NIOS Processor Image

Machine code can be loaded onto an image of the processor for which it was assembled and executed therein. In this case, the processor image is the same as a physical processor (for the most part).

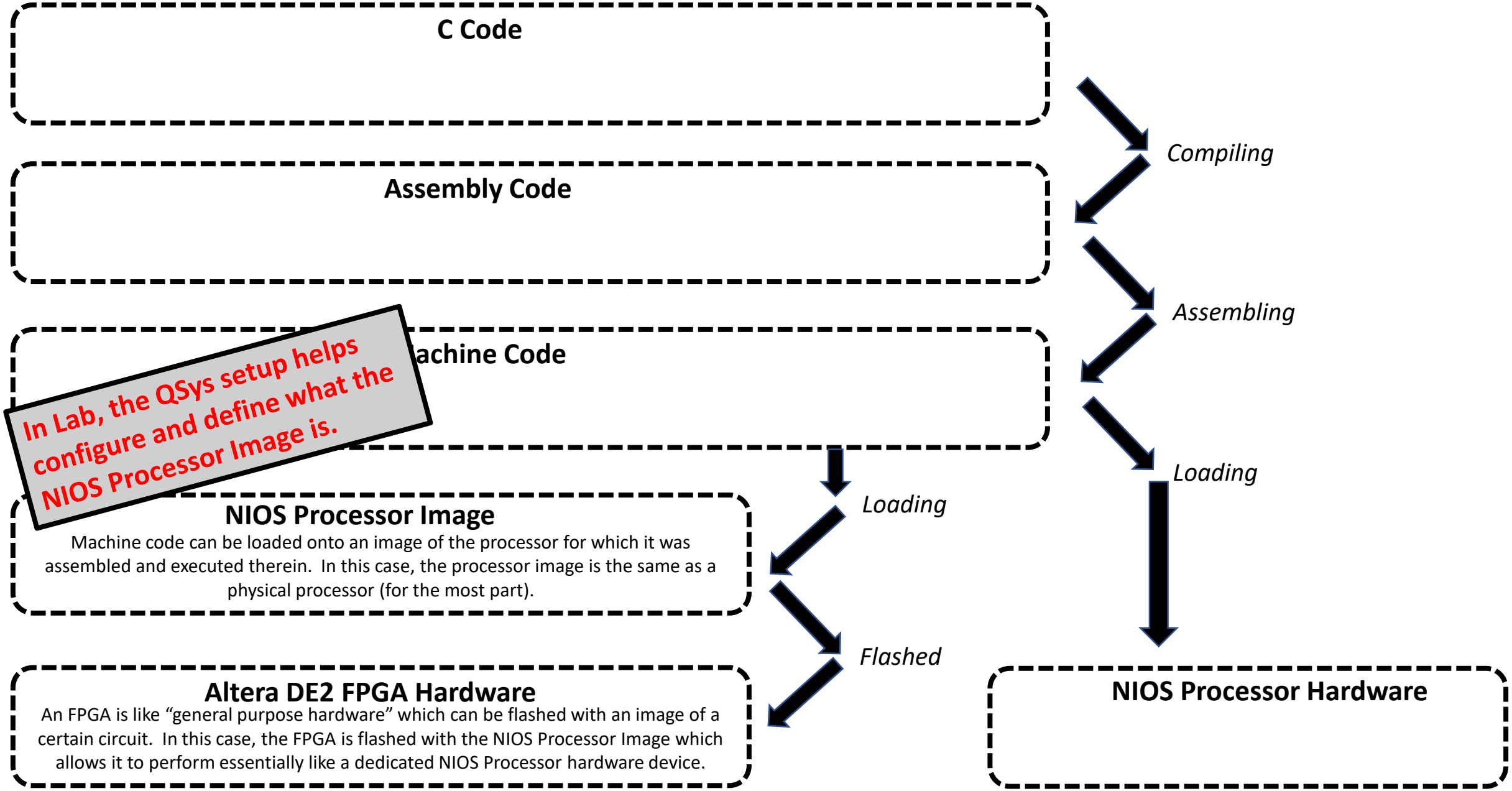
Flashed

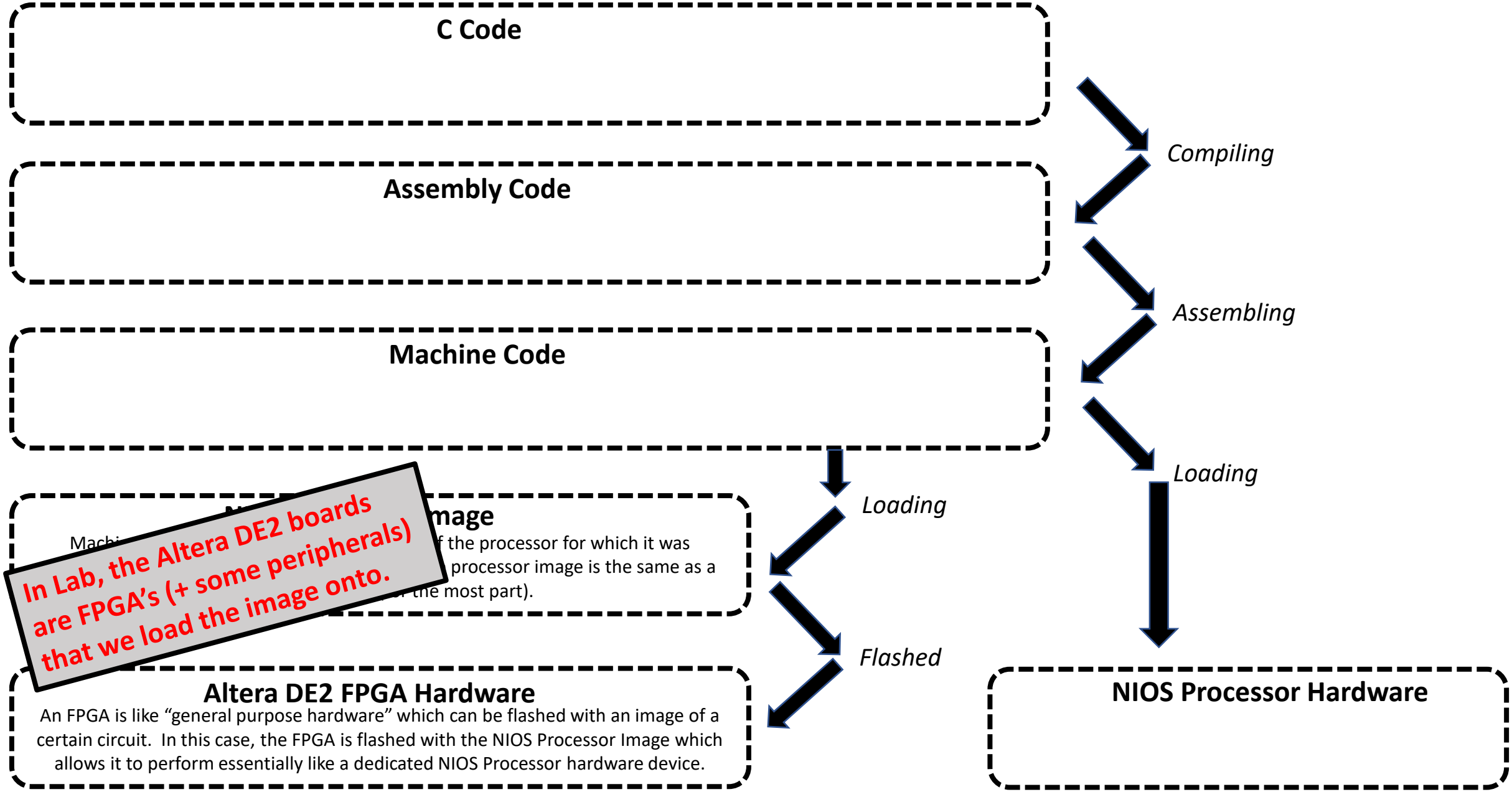
Altera DE2 FPGA Hardware

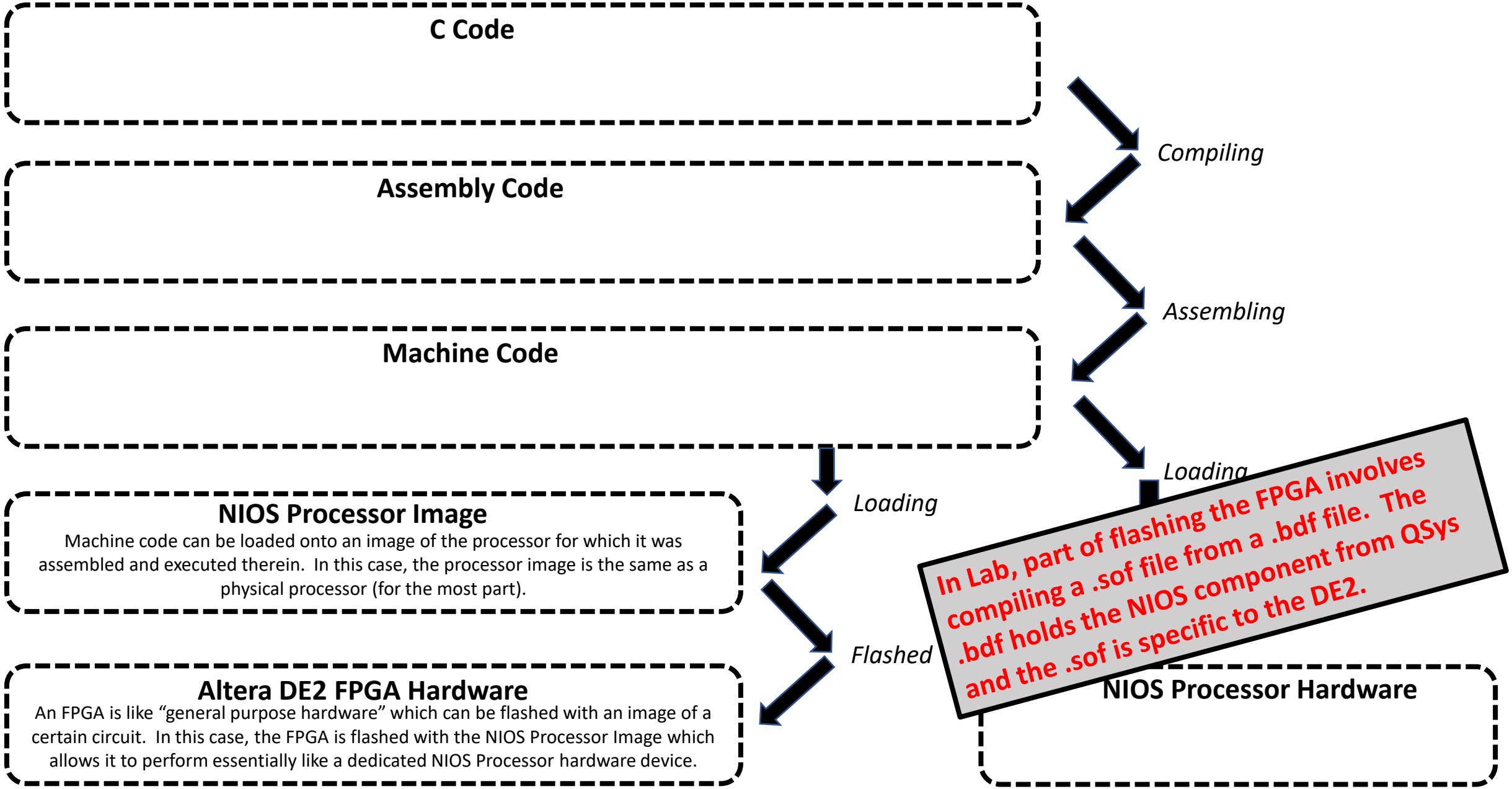
An FPGA is like "general purpose hardware" which can be flashed with an image of a certain circuit. In this case, the FPGA is flashed with the NIOS Processor Image which allows it to perform essentially like a dedicated NIOS Processor hardware device.

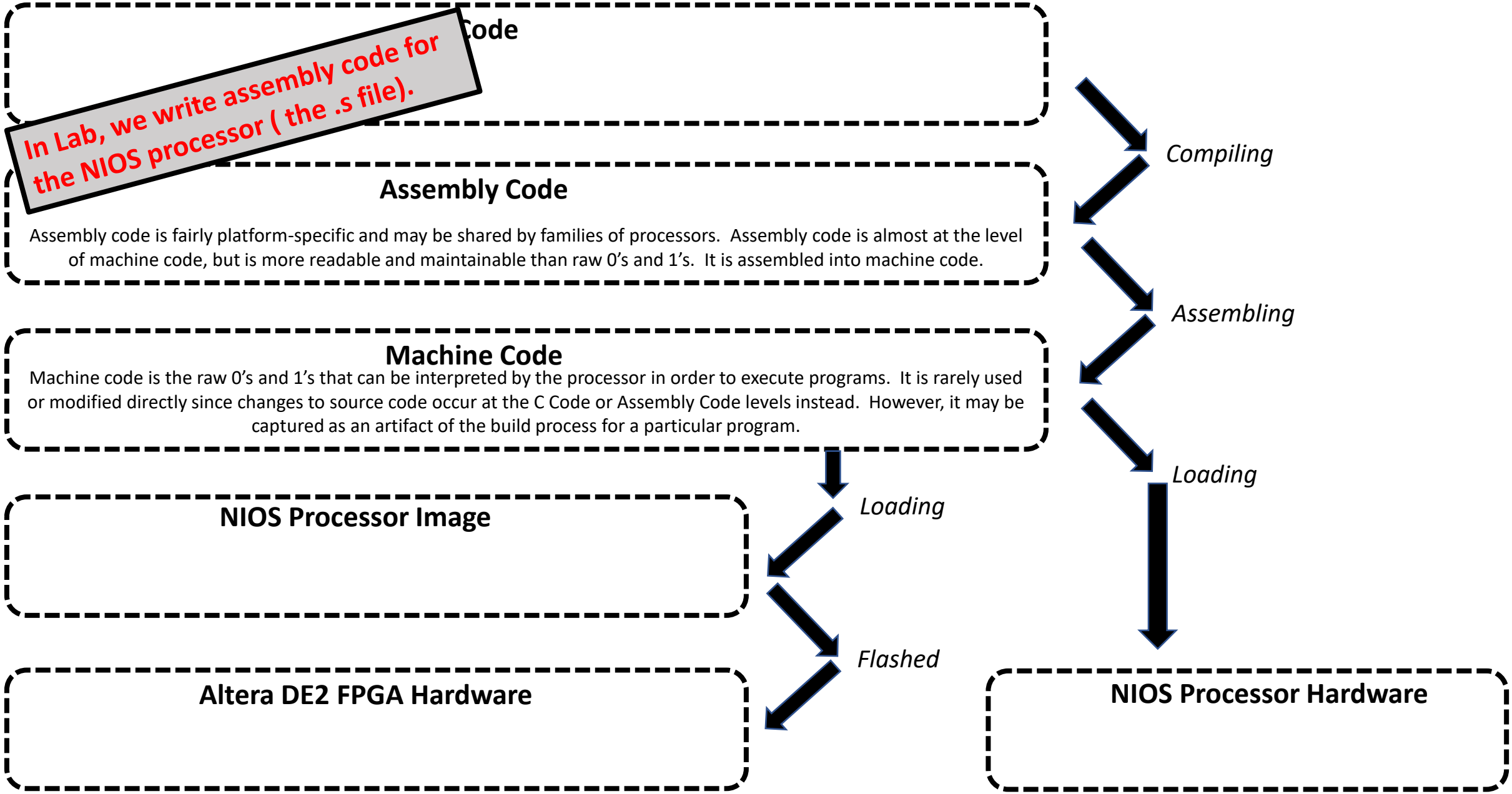
NIOS Processor Hardware

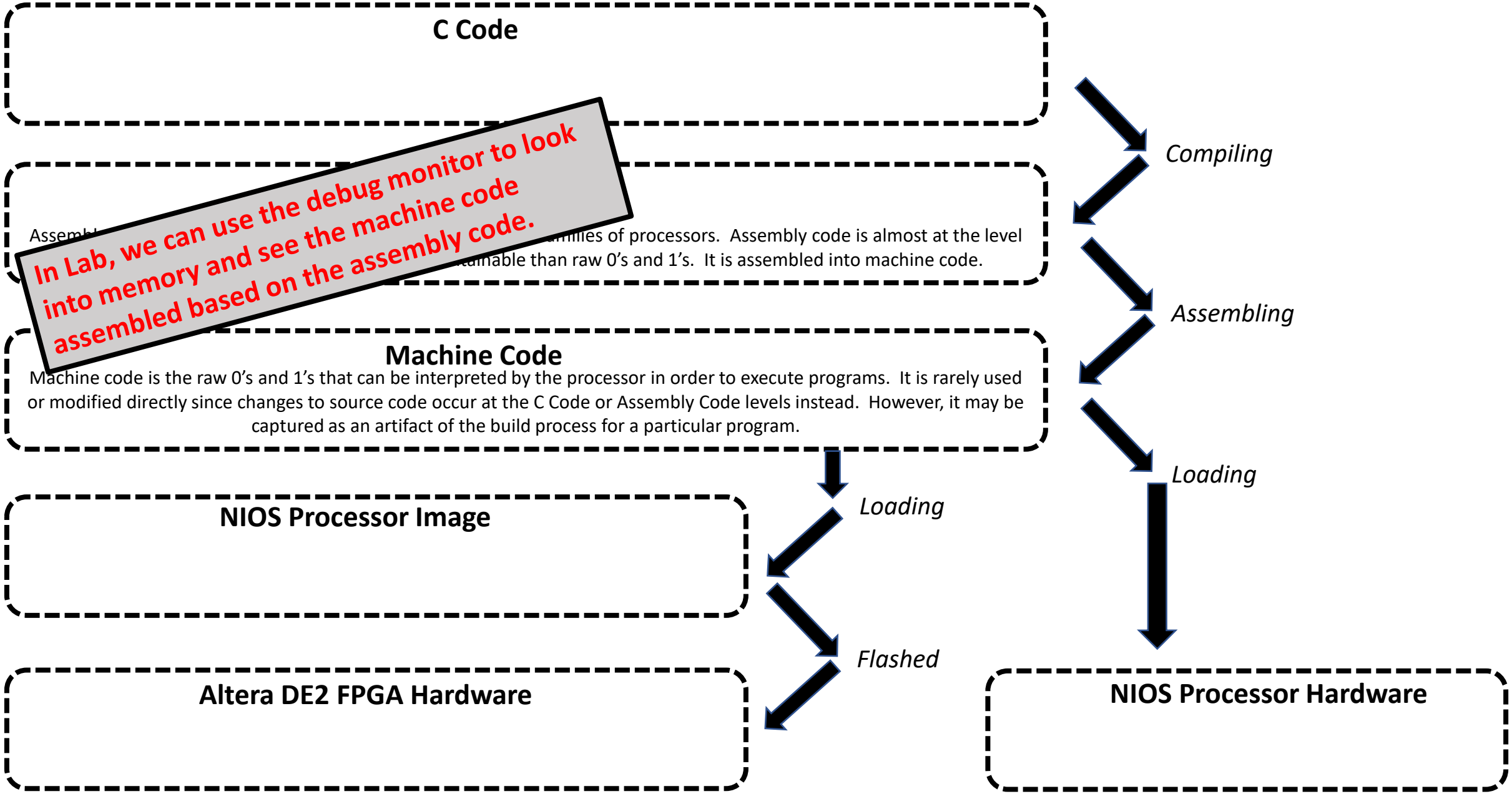
Machine code can be loaded onto the processor for which it was assembled and "physically" executed therein.

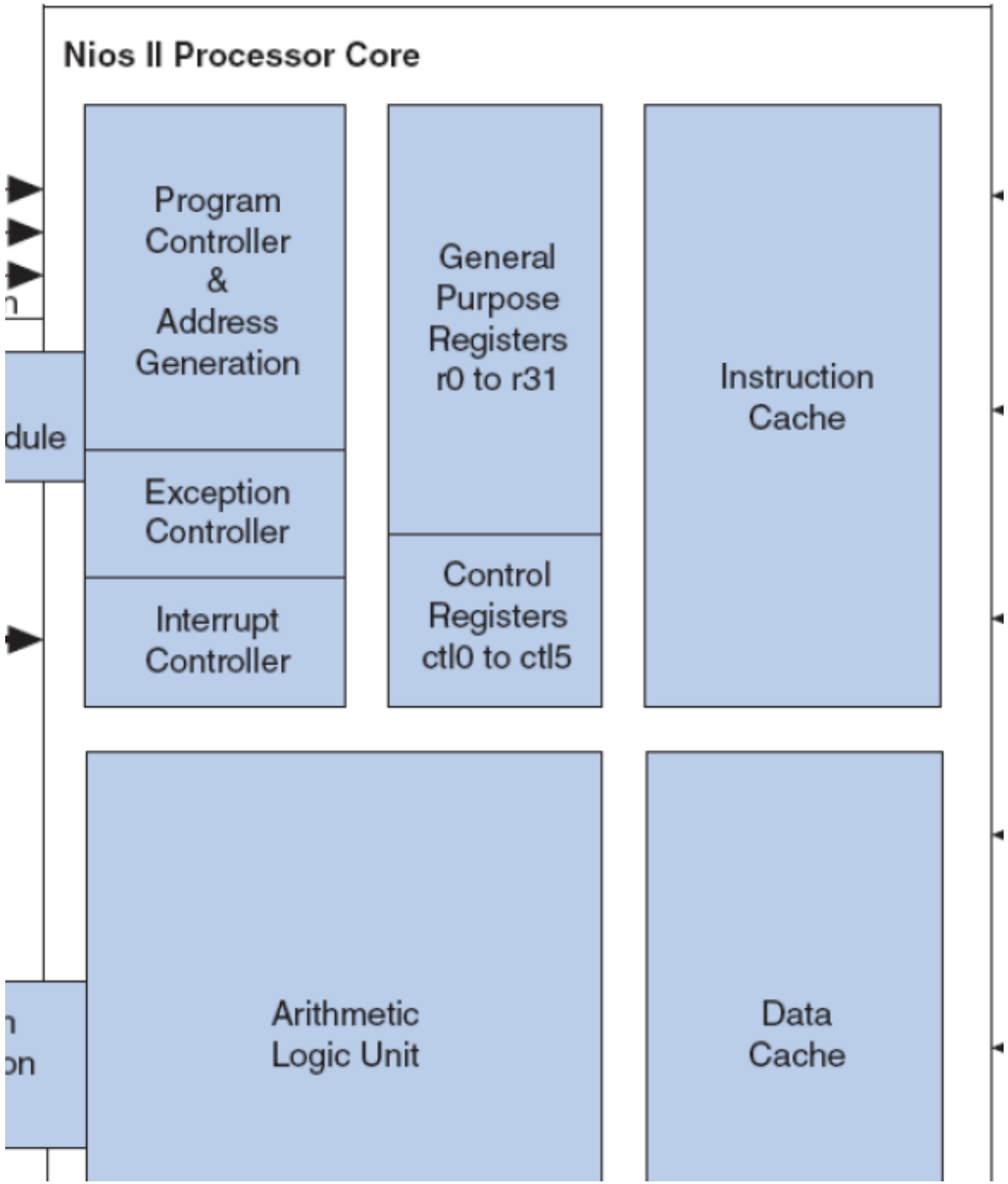

C Code
Compiling
Assembly Code
Assembling
achine Code
In Lab, the QSys setup helps configure and define what the NIOS Processor Image is.
Loading
Loading
NIOS Processor Image
Machine code can be loaded onto an image of the processor for which it was assembled and executed therein. In this case, the processor image is the same as a physical processor (for the most part).
Flashed
Altera DE2 FPGA Hardware
An FPGA is like “general purpose hardware” which can be flashed with an image of a certain circuit. In this case, the FPGA is flashed with the NIOS Processor Image which allows it to perform essentially like a dedicated NIOS Processor hardware device.
NIOS Processor Hardware


C Code
Compiling
Assembly Code
Assembling
Machine Code
Loading
Loading
mage
f the processor for which it was
processor image is the same as a
the most part).
In Lab, the Altera DE2 boards are FPGA's (+ some peripherals) that we load the image onto.
Flashed
Altera DE2 FPGA Hardware
An FPGA is like "general purpose hardware" which can be flashed with an image of a certain circuit. In this case, the FPGA is flashed with the NIOS Processor Image which allows it to perform essentially like a dedicated NIOS Processor hardware device.
NIOS Processor Hardware


C Code
Compiling
Assembly Code
Assembling
Machine Code
Loading
Loading
NIOS Processor Image
Machine code can be loaded onto an image of the processor for which it was assembled and executed therein. In this case, the processor image is the same as a physical processor (for the most part).
Flashed
Altera DE2 FPGA Hardware
An FPGA is like "general purpose hardware" which can be flashed with an image of a certain circuit. In this case, the FPGA is flashed with the NIOS Processor Image which allows it to perform essentially like a dedicated NIOS Processor hardware device.
In Lab, part of flashing the FPGA involves compiling a .sof file from a .bdf file. The .bdf holds the NIOS component from QSys and the .sof is specific to the DE2.
NIOS Processor Hardware


Code
In Lab, we write assembly code for the NIOS processor (the .s file).
Assembly Code
Assembly code is fairly platform-specific and may be shared by families of processors. Assembly code is almost at the level of machine code, but is more readable and maintainable than raw 0's and 1's. It is assembled into machine code.
Machine Code
Machine code is the raw 0's and 1's that can be interpreted by the processor in order to execute programs. It is rarely used or modified directly since changes to source code occur at the C Code or Assembly Code levels instead. However, it may be captured as an artifact of the build process for a particular program.
NIOS Processor Image
Altera DE2 FPGA Hardware
Compiling
Assembling
Loading
Loading
Flashed
NIOS Processor Hardware


C Code
Compiling
Assembl
milies of processors. Assembly code is almost at the level
inable than raw 0's and 1's. It is assembled into machine code.
In Lab, we can use the debug monitor to look into memory and see the machine code assembled based on the assembly code.
Assembling
Machine Code
Machine code is the raw 0's and 1's that can be interpreted by the processor in order to execute programs. It is rarely used or modified directly since changes to source code occur at the C Code or Assembly Code levels instead. However, it may be captured as an artifact of the build process for a particular program.
Loading
Loading
NIOS Processor Image
Flashed
Altera DE2 FPGA Hardware
NIOS Processor Hardware


Nios II Processor Core
Program Controller & Address Generation
Exception Controller
Interrupt Controller
General Purpose Registers r0 to r31
Control Registers ctl0 to ctl5
Instruction Cache
Arithmetic Logic Unit
Data Cache