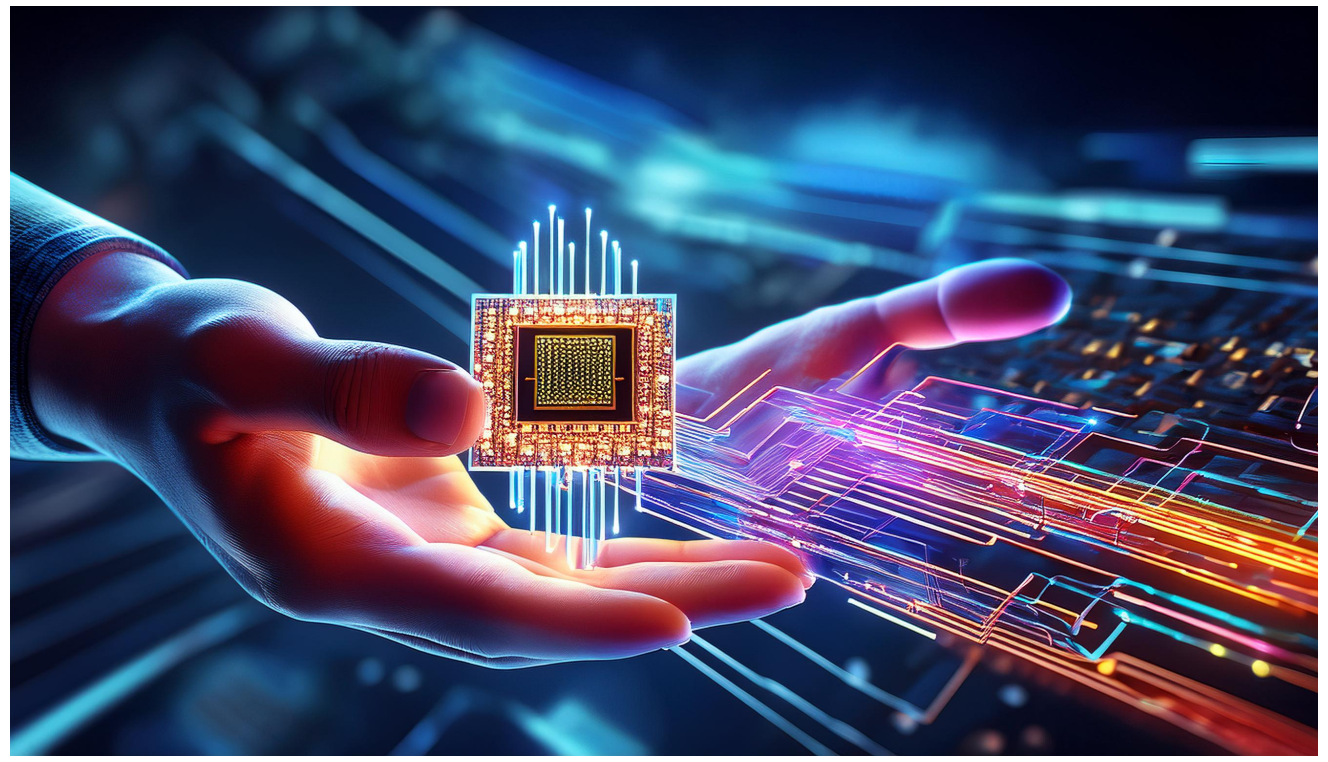
15.10.24 15:22

CSCI 250
Introduction to Computer Organisation
Lecture 3: CPU Architecture I



Jetic Gū

2024 Fall Semester (S3)

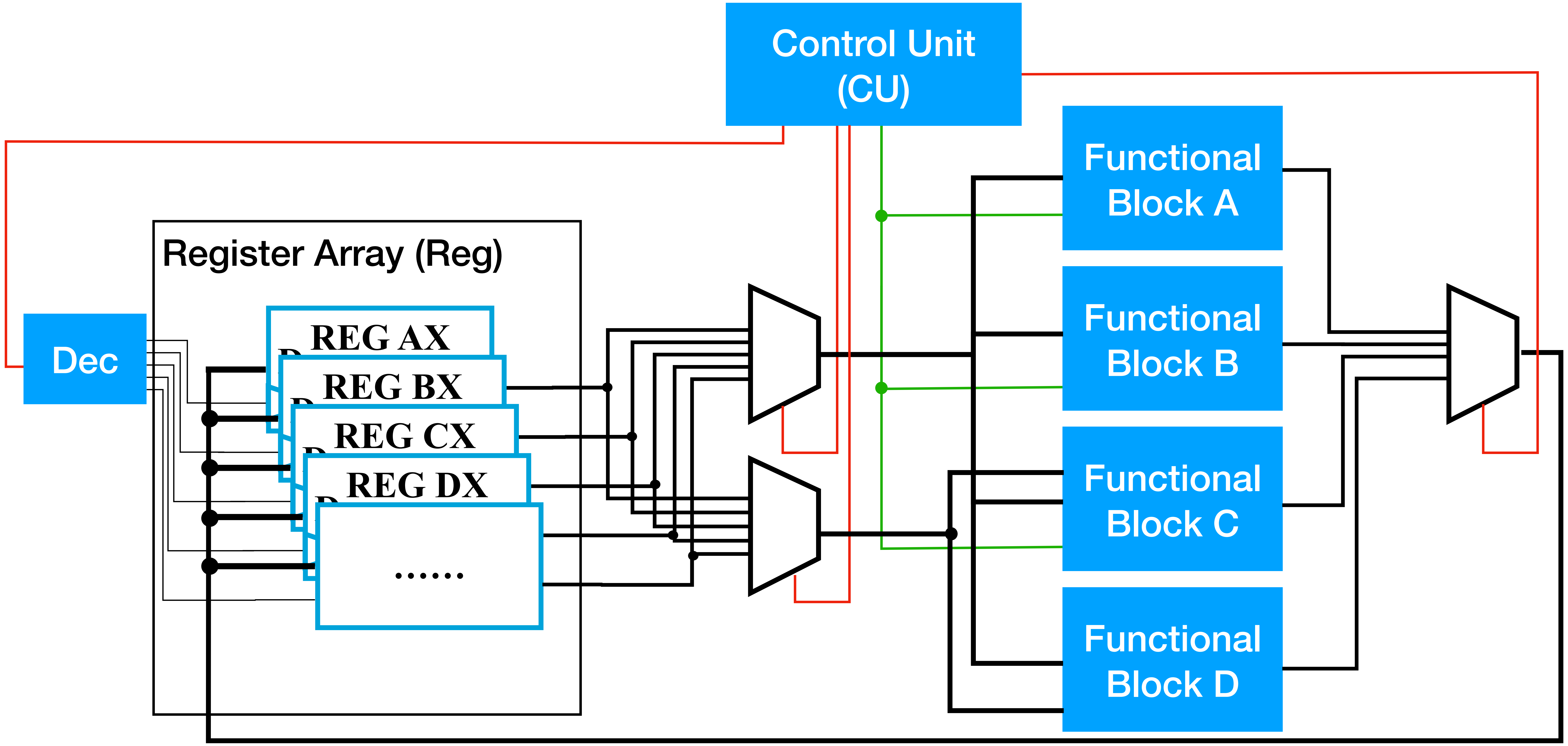
Overview

- Architecture: von Neumann
- Textbook: LCD: 9.7; CO: 2.1
- Core Ideas:
    1. Review
    2. Instructions

Datapath Review

P1 Review

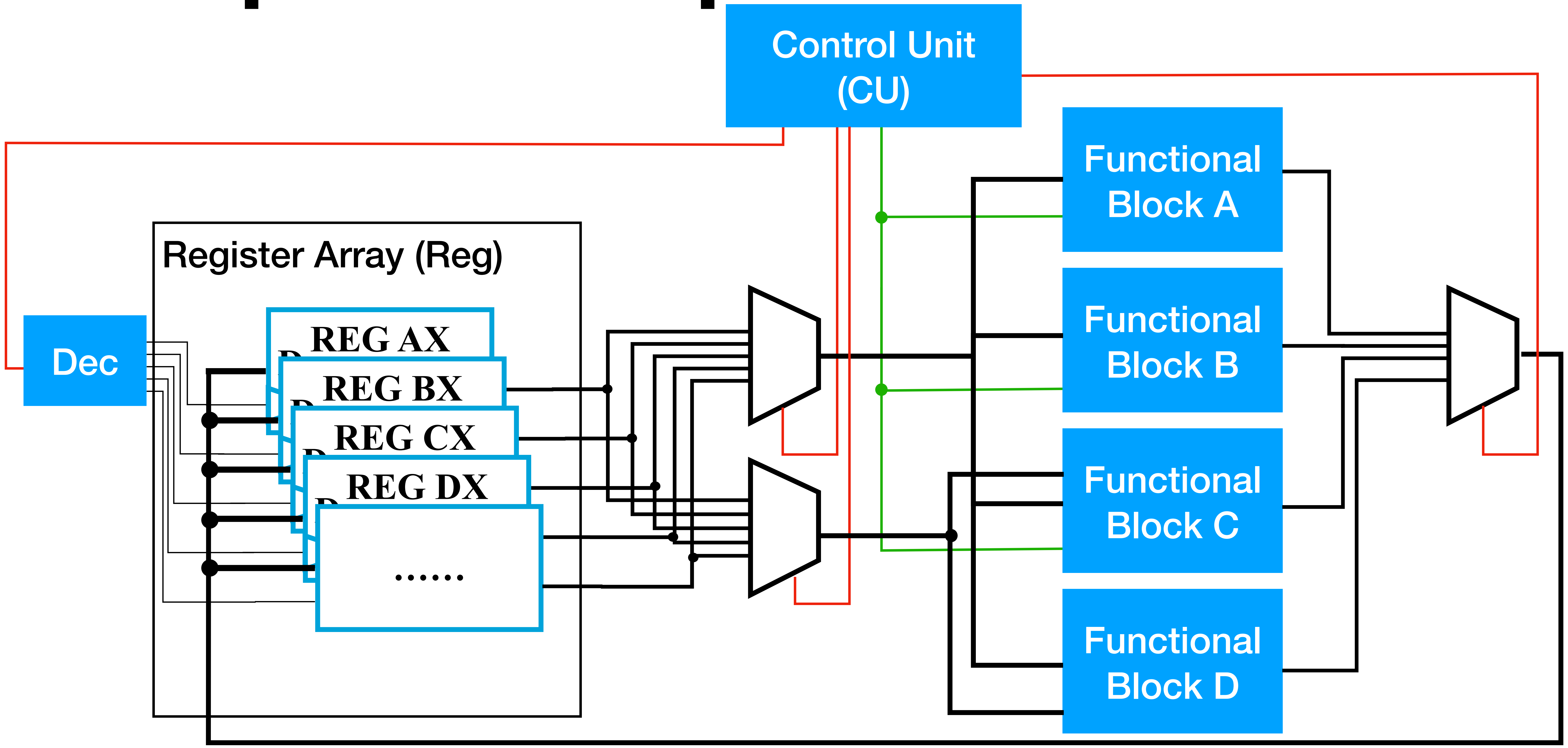
Example Datapath Architecture



Review

P1 Review

Example Datapath Architecture



- Register Array, MUX, DEC, etc.
- Functional Blocks: Arithmetic and Logical Unit (ALU), adder, subtractor, etc.

Review

P1 Review

Register Transfer Operations

	Operator	Example
Assignment	`<=`	`ax <= 12h`
Reg. Transfer	`<=`	`ax <= bx`
Addition	`+`	`ax + bx`
Subtraction	`-`	`ax - bx`
Shift Left	`sll`	`ax sll 2`
Shift Right	`srl`	`ax srl 2`

	Operator	Example
Bitwise AND	`and`	`ax and bx`
Bitwise OR	`or`	`ax or bx`
Bitwise NOT	`not`	`not ax`
Bitwise XOR	`xor`	`ax xor bx`
Vectors	`ax(3 down to 0)`	`ax(3 down to 0)`
Concatenate	`&`	`ax(7 down to 4)` `&ax(3 down to 0)`

Review

CPU Instructions

Words of a Computer

- A computer’s language: instructions
- A computer’s vocabulary: instruction set
- Instructions are pure binary code
- Instructions are CPU specific
  - CISC: 6502, M68k, x86, x86-64, etc.
  - RISC: PowerPC, ARM, etc.

Instructions of a Computer

- **Basic Register Micro-operations**: CSCI150
- **Data Transferring**: main memory
- **Jump** operations: go to specific instruction
  - Subroutine, `goto` expression, etc.
- **Conditional branch**: compare, if condition met go to specific instruction
  - `if`-triggered subroutine or `goto` expressions

Concept

P2
Instructions

Programming Languages

- C/C++: Compiled languages, requires a compiler
    - C programmes are the lowest level higher-level languages
    - The language of embedded systems, and OS kernel
    - Compiler "translates" C programmes to machine language in binary
    - Binary is not readable by human, so we use assembly as substitution

Technical

P2 Instructions

ARM 16bit Thumb Instructions

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OPCODE						Other operands									

- ARM: a family of RISC instruction set architectures (ISA)
  - Advanced RISC Machines
  - 32bit, 64bit
  - 16bit Thumb instruction set
    - A subset of instructions that might have restrictions, but for us it's good enough

Technical

P2
Instructions

ARM 16bit Thumb Instructions

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OPCODE						Other operands									

- OPCODE: operation code
  - **Instruction machine code/ instruction code**
    Portion of a machine language instruction that specifies the operation to be performed by the CPU

Technical

P2 Instructions

ARM 16bit Thumb Instructions

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OPCODE						Other operands									

Opcode	Instruction Encoding
`00xxxx`	Register/Immediate: Arithmetic Operations
`010000`	Register: Logical Operations
`010001`	Special Register data instructions*
`01001x`	Memory Load: from Literal Pool (PC with Offset)

`0101xx`
`011xxx`
`100xxx`
	Memory Load/Store (Single address)

`1010XX`	Relative Address calculation*
`1011xx`	Misc*
`1100xx`	Memory Load/Store (Blocks)*
`1101xx`	Conditional branch: `if`-triggered subroutine/`goto`
`11100x`	Unconditional branch: `jump`

Technical

P2
Instructions

ARM 16bit Thumb Instructions

- ARM Thumb instructions have access to 8 general purpose registers, although ARM-32 actually has 16 such registers
- Instructions are first stored in the main memory, then transferred to the CPU before it can be executed
    - Traditionally, an *instruction register* (not GPR) is used to store this
    - Modern CPUs for efficiency uses a special *instruction queue*
- A GPR keeps track of the address of the current instruction being executed
    - This is called the **Programme Counter** (PC), or R15 in ARM

Technical

P2 Instructions

ARM 16bit Thumb Instructions

CPU

PC	0000FFF0h
IR	

Main Memory

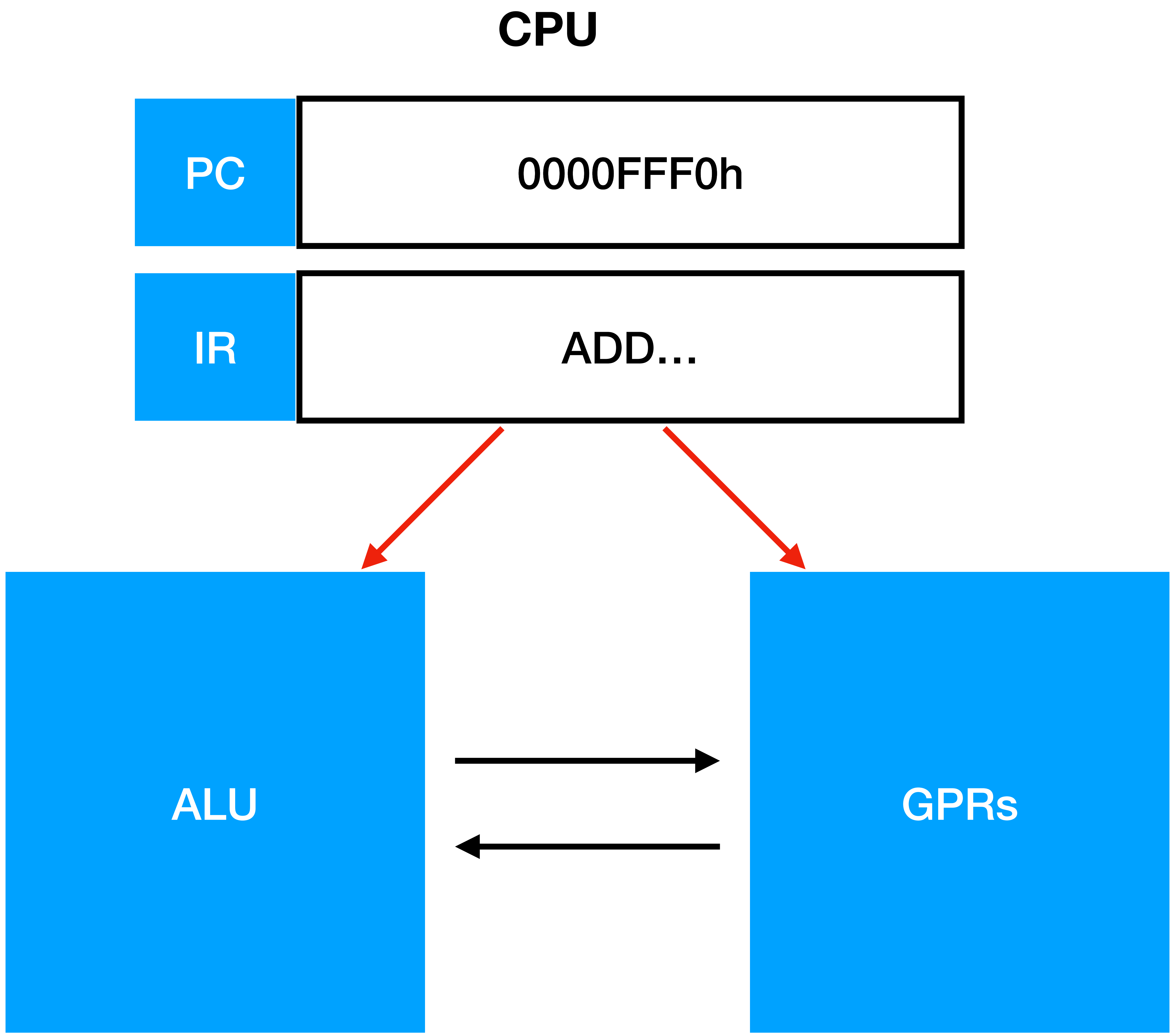
Address	Memory Content
0000FFF0h	**ADD…**
0000FFF2h	MOV…
0000FFF4h	ADD…
0000FFF6h	MOV…

1. Actual content may vary according to ARM specifications, this is just a simplified example

Technical

P2 Instructions

ARM 16bit Thumb Instructions



Main Memory

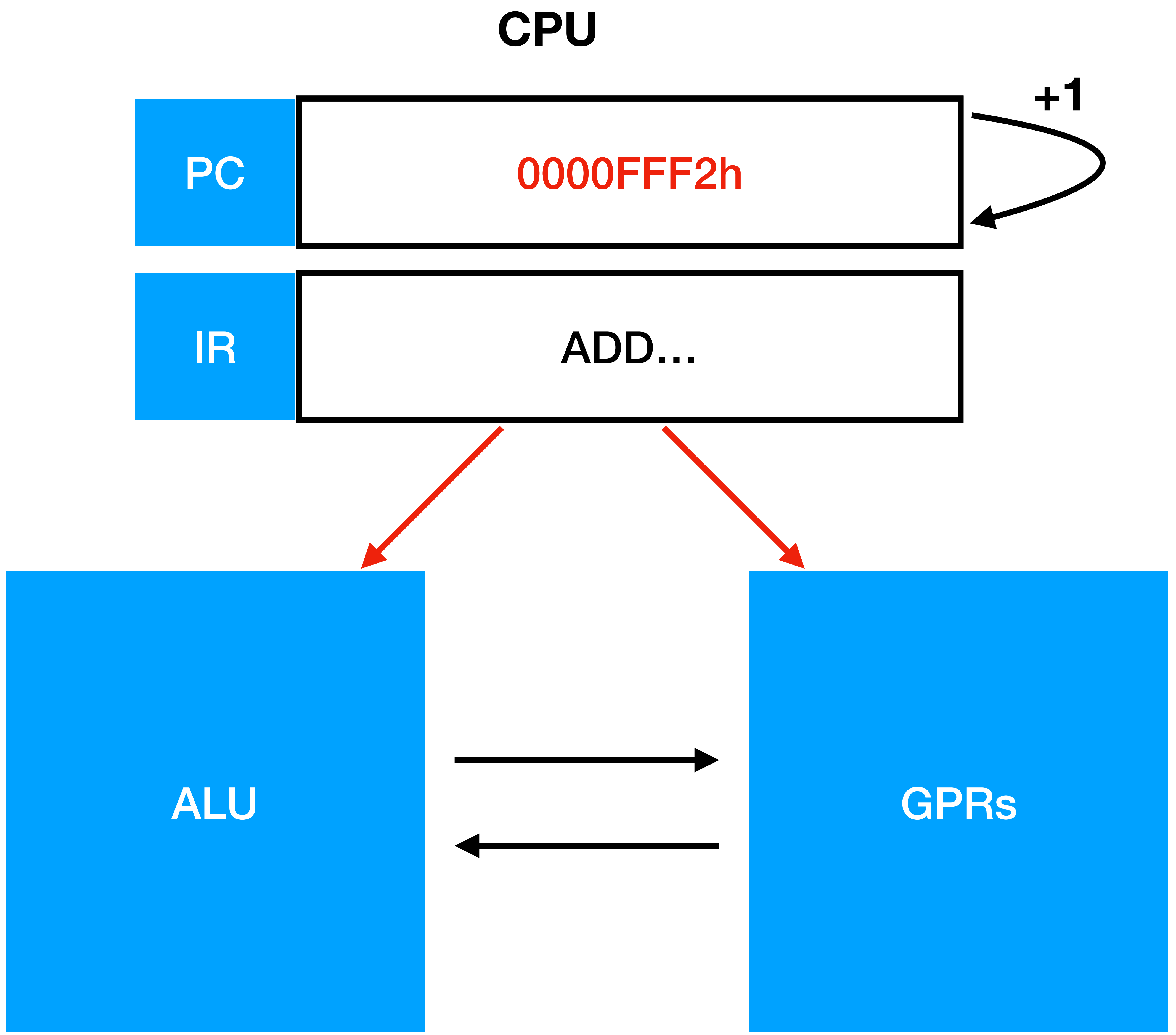
Address	Memory Content
0000FFF0h	ADD…
0000FFF2h	MOV…
0000FFF4h	ADD…
0000FFF6h	MOV…

Technical

1. Actual content may vary according to ARM specifications, this is just a simplified example

P2 Instructions

ARM 16bit Thumb Instructions



Main Memory

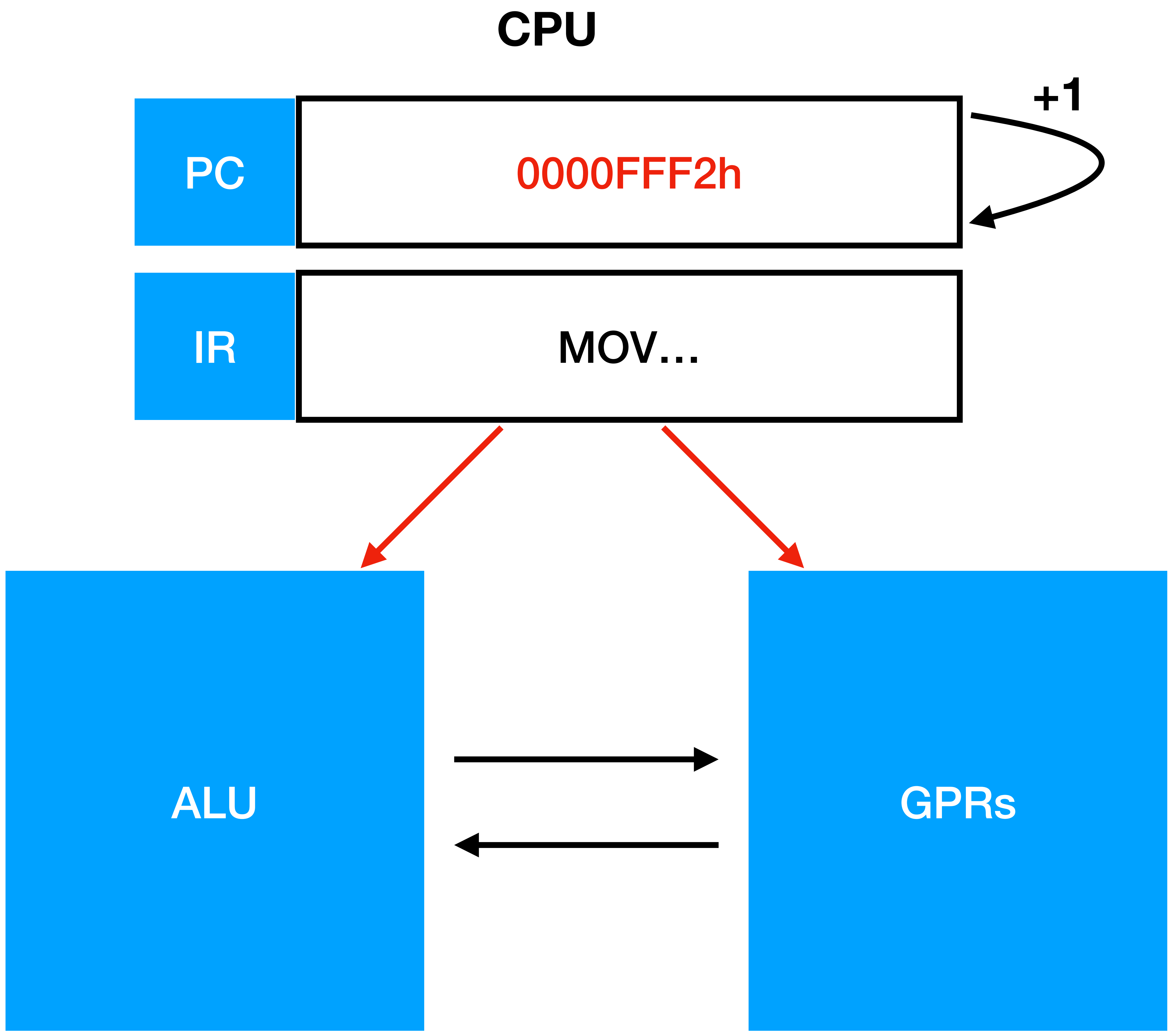
Address	Memory Content
0000FFF0h	ADD…
0000FFF2h	MOV…
0000FFF4h	ADD…
0000FFF6h	MOV…

1. At the next CLK tick, PC = PC + 1

Technical

P2
Instructions

ARM 16bit Thumb Instructions



Main Memory

Address	Memory Content
0000FFF0h	ADD…
0000FFF2h	MOV…
0000FFF4h	ADD…
0000FFF6h	MOV…

1. At the next CLK tick, PC = PC + 1

Technical

P2 Instructions

Instruction Register vs Queue

- Instruction Register
  - One instruction can be moved to the CPU at any time
  - After every instruction is executed, PC += 1, memory needs to be accessed so the next instruction could be brought in. This is very very slow, even with Cache
  - Speed things up: some CPUs can bring in new instructionss from memory when the current instruction is not performing memory access, so as to speed things up
  - Instruction specific cache: CPUs can have L1/L2 cache dedicated to instructions
- Instruction Queue
  - Intel Sandy Bridge (2009/2011): maintain a queue of instructions to be executed within the CPU, so no need to wait for memory access at the end of every instruction
  - Much much much faster

Concept