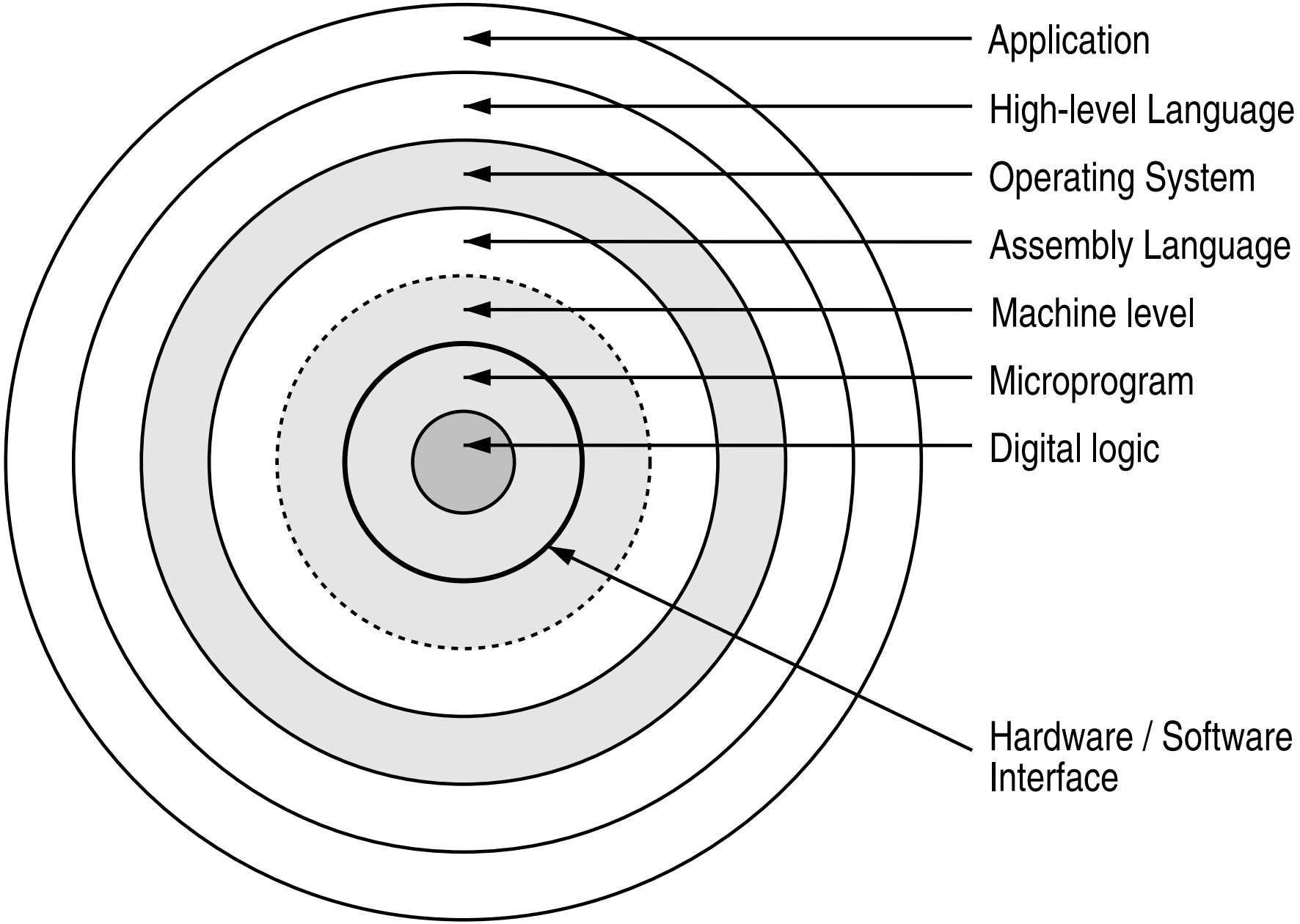
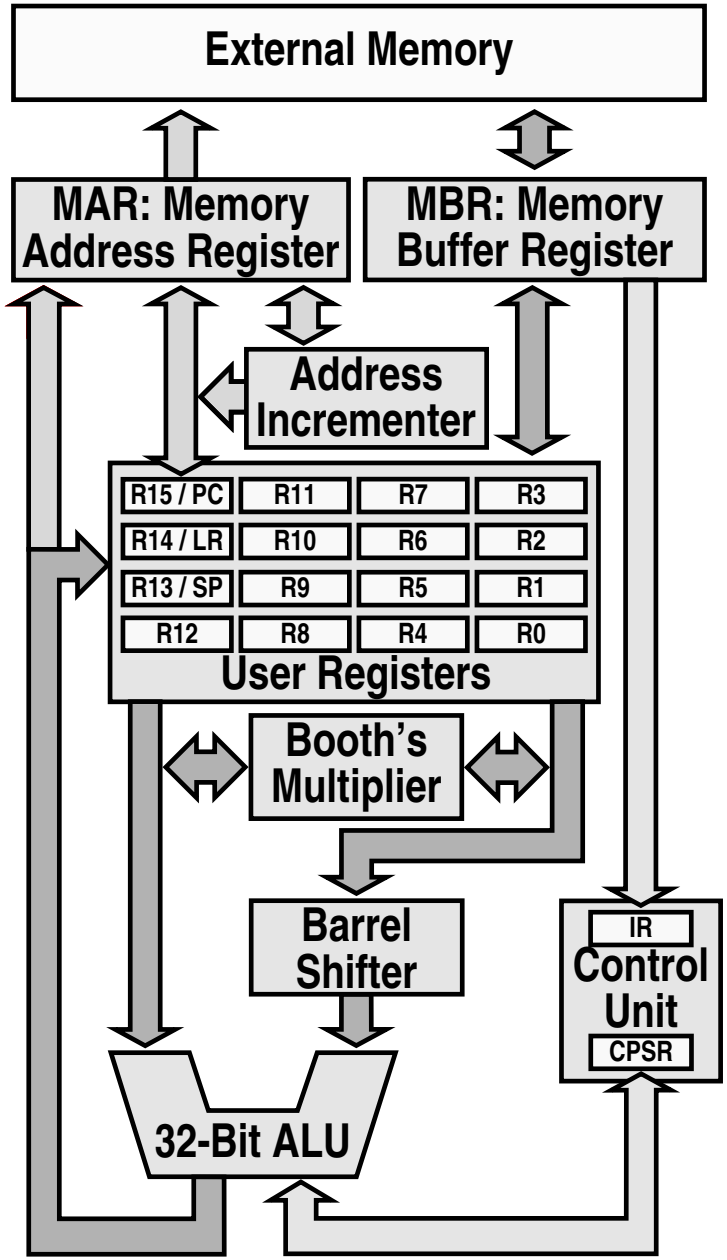
Systems Architecture

The Central Processing Unit

The Computer System



CPU Structure



CPU Registers

Internal Registers	
PC	Program Counter
IR	Instruction Register
MAR	Memory Address Register
MBR	Memory Buffer Register
CPSR	Current Processor Status Register

Condition Flags	
C	Carry
Z	Zero
N	Negative
V	Overflow

Internal Devices	
ALU	Arithmetic Logic Unit
CU	Control Unit
M	Memory Store
MMU	Mem Management Unit

User Registers	
Rn	Register n $n = 0 \ldots 15$
SP	Stack Pointer
LR	Link Register

Note that each CPU has a different set of User Registers

Current Process Status Register

- Holds a number of status *flags*:

  N True if result of last operation is Negative

  Z True if result of last operation was Zero or equal

  C True if an unsigned borrow (Carry over) occurred
  Value of last bit shifted

  V True if a signed borrow (oVerflow) occurred

- Current execution *mode*:

  User Normal “user” program execution mode

  System Privileged operating system tasks
  Some operations can only be preformed
  in a System mode

Register Transfer Language

NAME	Value of register or unit
$\leftarrow$	Transfer of data
x:	Guard, only if x true
($field$)	Specific $field$ of unit
	($name$), bit (n) or range (n:m)
Rn	User Register n
num	Decimal number
2_num	Binary number
0xnum	Hexadecimal number
M($addr$)	Memory Access ($addr$)
IR($field$)	Specified $field$ of IR
ALU($field$)	Specified $field$ of the Arithmetic and Logic Unit

MAR $\leftarrow$ PC

$\langle cc \rangle$: MAR $\leftarrow$ PC

ALU(C) $\leftarrow$ 1

R0 $\leftarrow$ MBR(0:7)

R0 $\leftarrow$ MBR

R0 $\leftarrow$ 128

R1 $\leftarrow$ 2_0100 0001

R2 $\leftarrow$ 0x40

MBR $\leftarrow$ M(MAR)

CU $\leftarrow$ IR(op-code)

ALU(C) $\leftarrow$ 1

Control Unit

- Controls operation of CPU
- Decodes op-code field of IR
- Two methods of implementation:

Microcode:

- Fairly slow
- Easy to design
- Very flexible

Random Logic:

- Very fast
- Difficult to design
- Fixed design

Control Unit Implementation

- **Microcode**
  CPU within a CPU
  Execute RTL-like microinstructions
  op-code is pointer to microcode program
  Has own microprogram ROM and CU
  microCU implemented in random logic

- **Random Logic**
  Not Random rather different for each design
  op-code decoded directly by boolean logic
  All CPU control via boolean logic using
  control, data, and address busses

Assembler Code Terminology

Label	*Mnemonic / Directive*	*Operands*	*Comment*
`Main`	`MOV`	`r0, #0`	`; move 0 into R0`

label Give a name to the location of the instruction

mnemonic Human readable name given to an instruction
`MOV` (Move) or `LDR` (Load Register)

operands Arguments for a given instruction
effective address (Data or Memory)

directive Instructions to the assembler
`END` (End of program source)

Fetch / Execute Cycle

To perform a function the CPU must first *fetch* the instruction from the main store before it can *execute* it.

For the instruction `adds r0, r1, #2` the CPU would execute:

FETCH	MAR ⟵ PC	move contents of PC to MAR
	PC ⟵ PC + 4	increment contents of PC
	MBR ⟵ M(MAR)	read instruction from memory
	IR ⟵ MBR	move instruction to IR
	CU ⟵ IR(op-code)	move op-code from IR to CU
EXECUTE	ALU(0) ⟵ R1	load ALU from R1
	ALU(1) ⟵ IR(op2)	load ALU from second operand
	ALU(cmd) ⟵ add	ask ALU to add values
	R0 ⟵ ALU(ans)	copy answer to R0
⟨*S*⟩:	CPSR ⟵ ALU(flags)	copy flags to PSR

Instruction Format

- All instructions include an operation code (op-code) field
- Instructions fall into different “groups”
- Each group of instructions has a different format
- Instructions groupings in order of use:

Data Movement	45.28%	Logical	3.91%
Flow Control	28.73%	Shift	2.92%
Arithmetic	10.75%	Bit Manipulation	2.05%
Compare	5.92%	I/O & Others	0.44%