06 The Execution Cycle

CSC 230

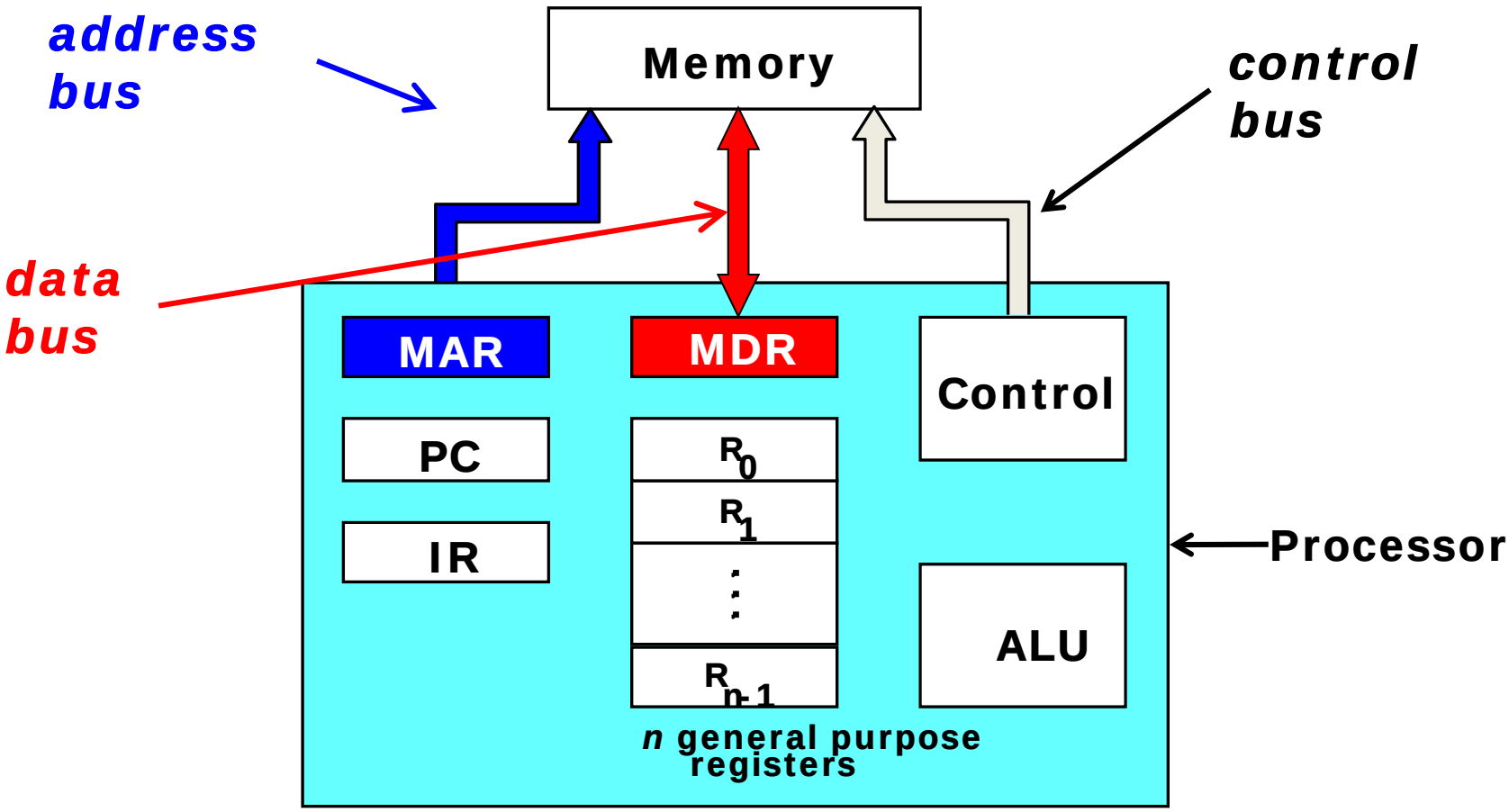
CPU activity follows the: FETCH - DECODE – EXECUTE - STORE sequence

FETCH - DECODE - EXECUTE

1. fetch (load) instruction from memory
2. *update current pointer to instructions memory address*
3. decode instruction
4. *fetch further data, if required*
5. execute instruction
6. go to step (1)

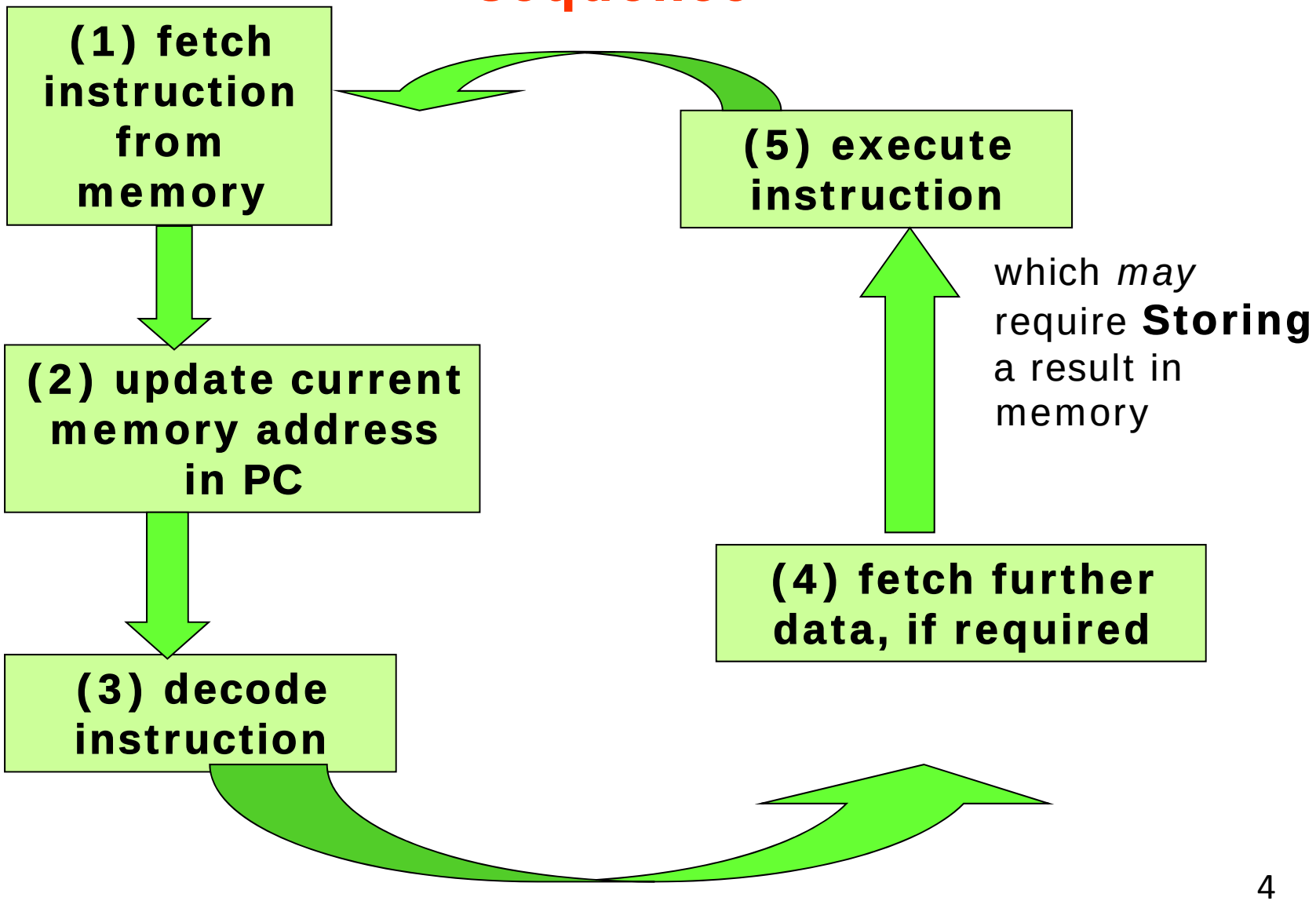
Registers

PC	Program counter
IR	Instruction register
MAR	Memory Address Register
MDR	Memory Data Register
AC	Accumulator

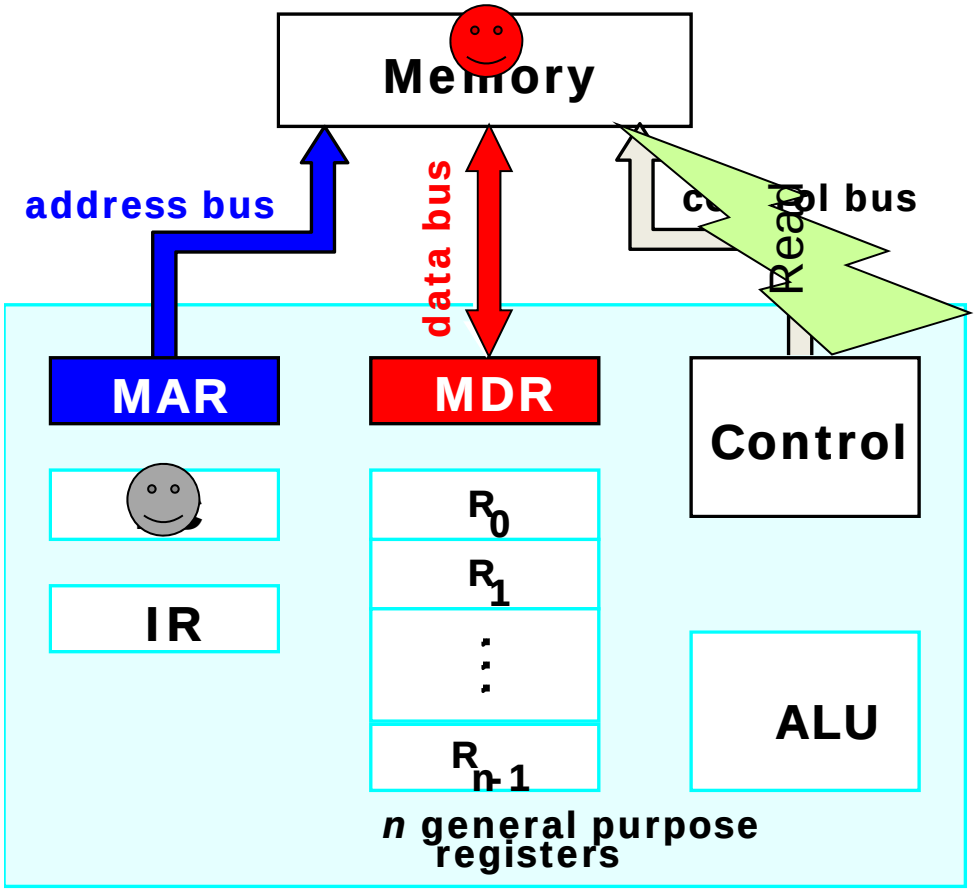


Connections between the processor and the memory

CPU activity follows the: FETCH - DECODE – EXECUTE - STORE sequence



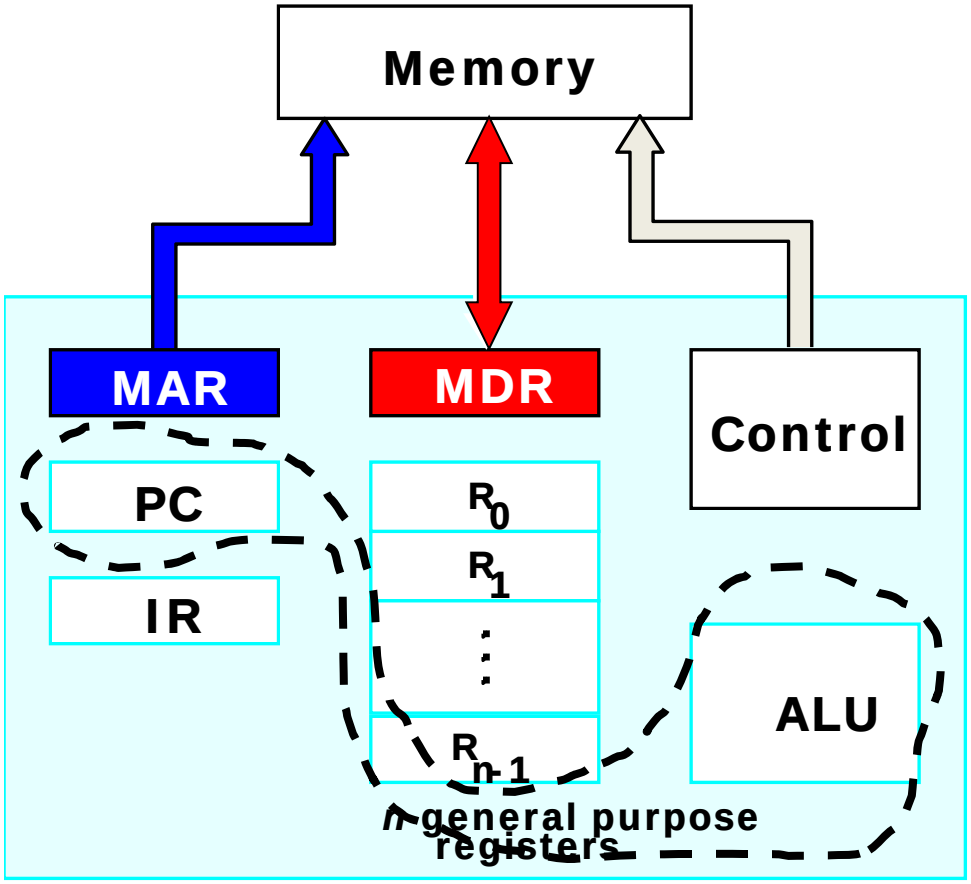
Detailed CPU activity



(1) fetch instruction from memory

a) address in PC → MAR
b) MAR → Address Bus
c) read signal → Control Bus
d) Wait for memory
e) content of location in memory → Data Bus
f) Data Bus → MDR
g) MDR → IR

Detailed CPU activity



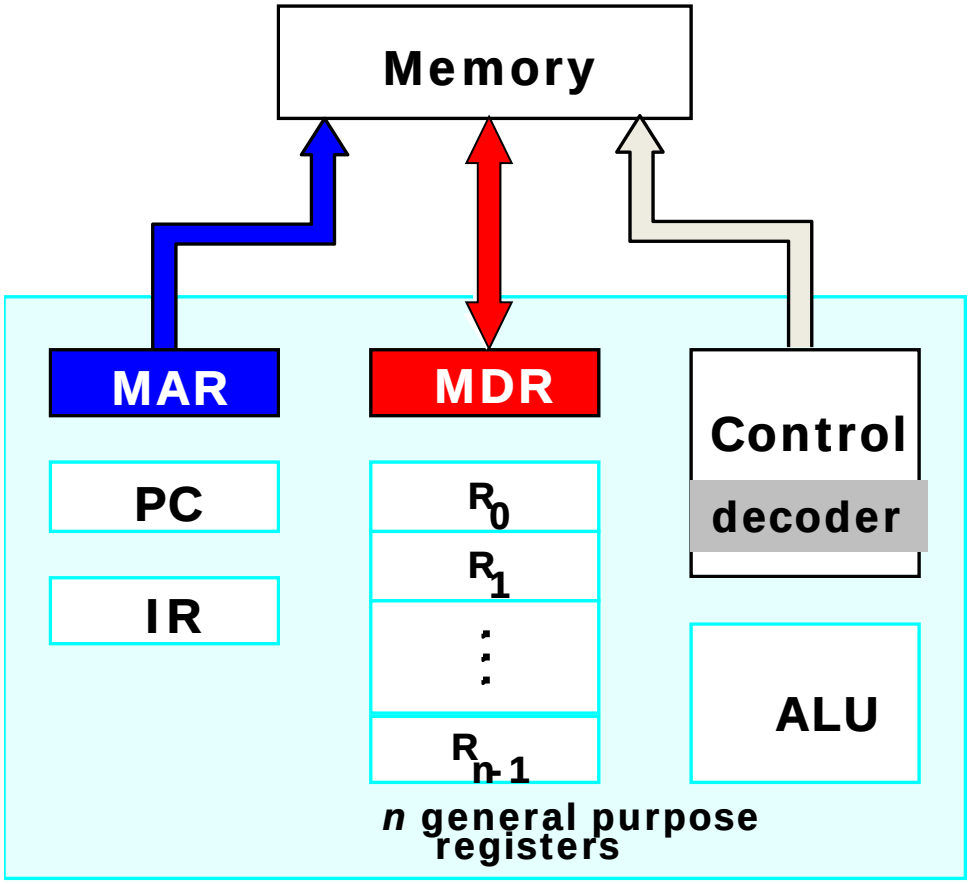
(2) update current memory address

a) increment PC

(ready for next instruction)

➔ Use the ALU

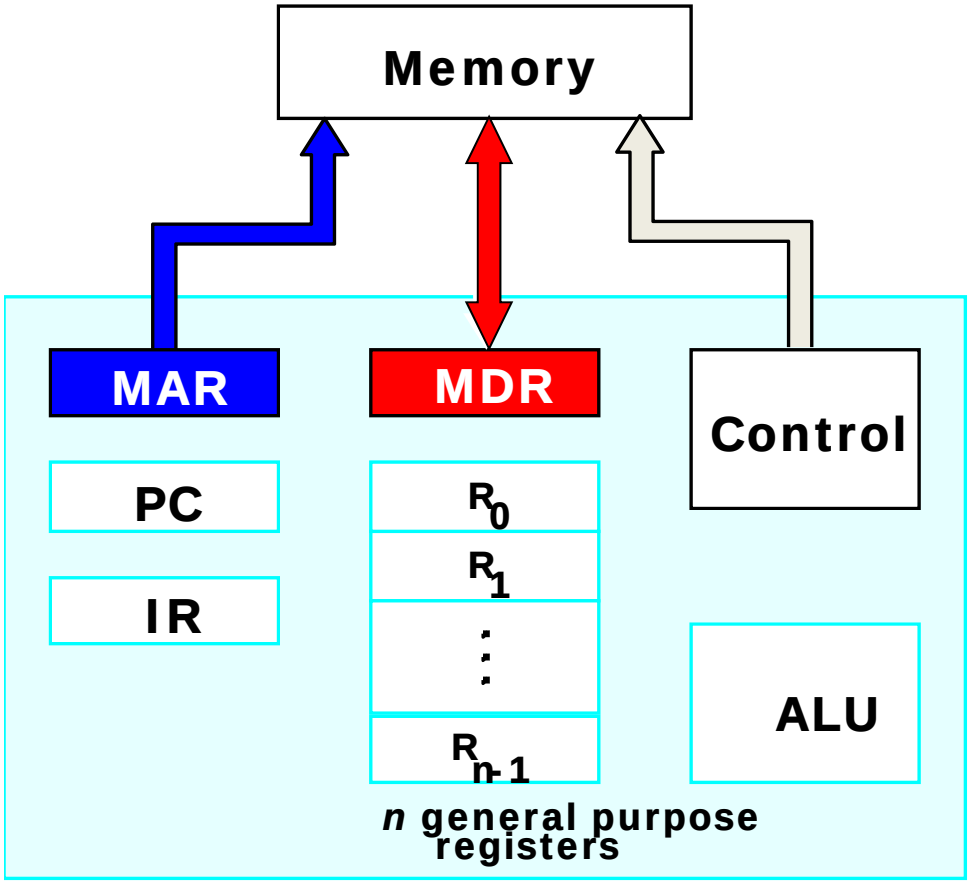
Detailed CPU activity



(3) decode instruction

- decode phase looks at opcode and operands in the content of the IR
- "Decoder" is a hardware component in the Control unit of the CPU

Detailed CPU activity

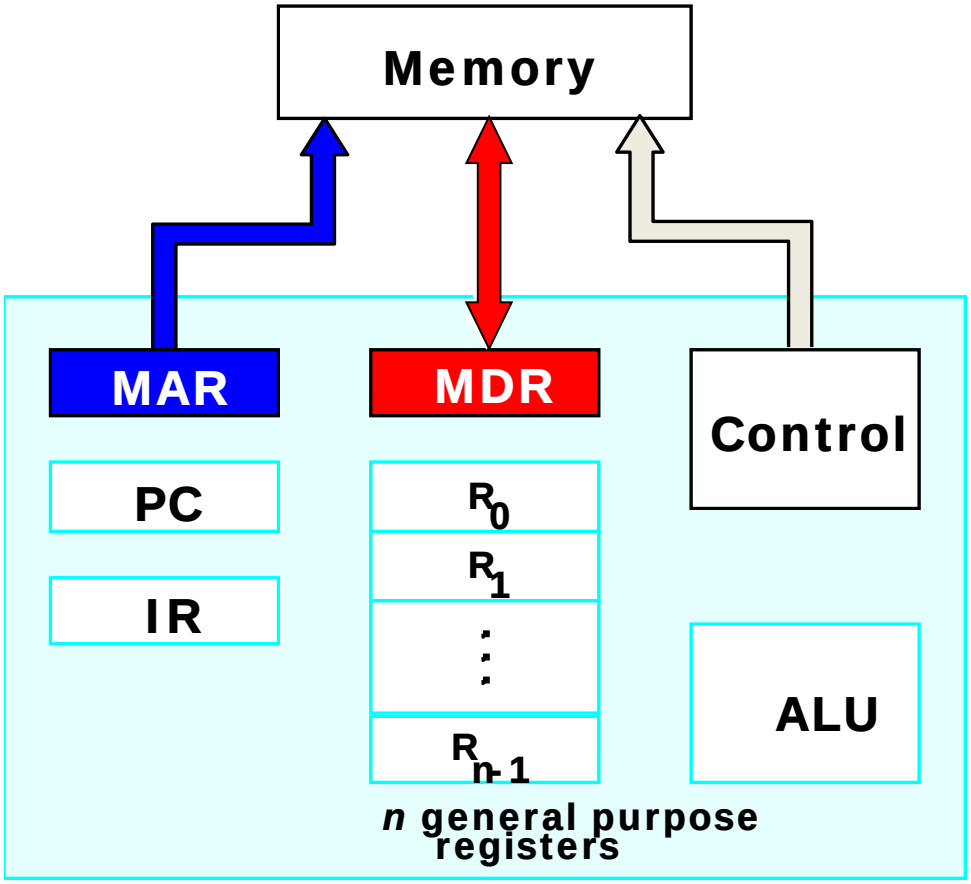


(4) fetch further data, if required

MAR, MDR and PC are possibly in use

There could be more Read cycles to fetch operands

Detailed CPU activity



(5) execute instruction

May involve ALU, plus reads and writes to memory

Detailed CPU activity

1. **fetch** instruction from memory
   - address in PC → MAR
   - MAR → Address Bus
   - read signal → Control Bus
   - Wait for memory
   - content of location in memory → Data Bus
   - Data Bus → MDR
   - MDR → IR
2. **update** current memory address
   - increment PC (ready for next instruction)
3. **decode** instruction
   - decode phase looks at opcode and operands
4. **fetch further data,** if required
   - (MAR, MDR and PC are possibly in use)
5. **execute** instruction
   - (may involve ALU, reads and writes to memory)
6. go to step (1)

Machine Language Execution Notes

Each machine language instruction → operation code (opcode)

Each machine language instruction → zero or more operands

Operands = the data, or the address of the data, to be used in the operation

Opcode ➔ may contain implicit operand

Each machine language instruction → occupies some number of ***consecutive*** bytes of memory

Each machine program → consists of a sequence of instructions stored consecutively in memory and executed in sequence

Program Counter (PC) → contains *the address of the next instruction code to be fetched*

The Instruction Execution Cycle: First (ARM) Examples

MOVE: copy a constant to a register, or copy between registers

MOV R1,#12	**copy the constant value 12 into register R1**
MOV R2,R5	**copy the contents of R5 into R2**

ADD: add the contents of two registers and assign the result to a register

ADD R3,R1,R2	**add the contents of R1 to R2 and assign the result to R3**

LOAD: copy the contents of a location in memory to a register

LDR R2,[R5]	**copy the content of memory, at address given by the content of R5, to register R2**

STORE: copy the content of a register into a location in memory

STR R3,[R4]	**copy the content of R3 into memory at address given by the content of R4**

The example and the opcodes

Memory address	Assembly language	Machine code
0x0000C000	MOV R1,#12	E3 A0 10 0C
0x0000C004	LDR R2,[R5]	E5 95 20 00
0x0000C008	ADD R3,R1,R2	E0 81 30 02
0x0000C00C	STR R3,[R4]	E5 84 30 00

Small example:

- ✓ Assume the program is loaded into memory and starts at address 0x0000C000
- ✓ Examine the Fetch/Decode/Execute Cycle in details

Instruction 1: steps (1) and (2)

Memory address	Assembly language	Machine code
0x0000C000	**MOV R1,#12**	**E3 A0 10 0C**

(1) fetch instruction from memory

a) PC is = 0x0000C000

b) MAR ← PC = 0x0000C000

c) 0x0000C000 placed on address bus by MAR

d) CPU issues READ command to memory using a signal on the Control bus

e) Wait for memory to respond

f) 32-bit content of location 0x0000C000 =

= 0xE3A0100C is returned on data bus to MDR

(2) update current memory address

a) IR ← MDR and 0xE3A0100C placed in the instruction register (IR)

b) PC is incremented by 4 to become 0x0000C004

Instruction 1: step (3)

Memory address	Assembly language	Machine code
0x0000C000	**MOV R1,#12**	**E3 A0 10 0C**

(3) decode instruction

a) Decoder in control unit recognizes MOV instruction

b) Addressing mode is “immediate” ➔ constant is inside the instruction opcode itself

c) No more READ cycles to memory needed

d) PC unchanged

Instruction 1: step (4)

Memory address	Assembly language	Machine code
0x0000C000	**MOV R1,#12**	**E3 A0 10 0C**

(4) execute instruction

a) R1 gets the content “12” copied from the IR directly

b) PC unchanged

Instruction 2: steps (1) and (2)

Memory address	Assembly language	Machine code
0x0000C004	**LDR R2,[R5]**	**E5 95 20 00**

(1) fetch instruction from memory

a) PC is = 0x0000C004

b) MAR ← PC = 0x0000C004

c) 0xC004 placed on address bus by MAR

d) CPU issues READ command to memory using a signal on the Control bus

e) Wait for memory to respond

f) 32-bit content at location 0x0000C004 =

= 0xE5952000 is returned on data bus to MDR

(2) update current memory address

a) IR ← MDR and 0x E5952000 placed in the instruction register (IR)

b) PC is incremented by 4 to become 0x0000C008

Instruction 2: step (3)

Memory address	Assembly language	Machine code
0x0000C004	**LDR R2,[R5]**	**E5 95 20 00**

(3) decode instruction

a) Decoder recognizes LDR instruction

b) Addressing mode is “indirect” ➔ the content of R5 is the *address* of required location in memory (that is, R5 is a “pointer”)

c) Need to have a READ cycle to get the *value* from memory

Instruction 2: step (4)

Memory address	Assembly language	Machine code
0x0000C004	**LDR R2,[R5]**	**E5 95 20 00**

(4) execute instruction

a) Need to have a READ cycle to get the value from memory

b) MAR ← R5 i.e. content of R5 copied to MAR and then placed on address bus by MAR

c) CPU issues READ command to memory using a signal on the Control bus then waits for memory to respond

d) 32-bit content of memory at address= R5 is returned on data bus to MDR

e) R2 ← MDR and R2 gets the content of memory just fetched, that is, the content of the location in memory whose address was in R5

f) PC unchanged

Instruction 3: steps (1) and (2)

Memory address	Assembly language	Machine code
0x0000C008	**ADD R3,R1,R2**	**E0 81 30 02**

(1) fetch instruction from memory

a) PC is = 0x0000C008

b) MAR ← PC = 0x0000C008

c) 0xC008 placed on address bus by MAR

d) CPU issues READ command to memory using a signal on the Control bus

e) Wait for memory to respond

f) 32-bit content at location 0x0000C008 =

= 0xE0813002 is returned on data bus to MDR

(2) update current memory address

a) IR ← MDR and 0x E0813002 placed in instruction register (IR)

b) PC is incremented by 4 to become 0x0000C00C

Instruction 3: step (3)

Memory address	Assembly language	Machine code
0x0000C008	**ADD R3,R1,R2**	**E0 81 30 02**

(3) decode instruction

a) Decoder in control unit recognizes ADD instruction

b) No more READ cycles to memory needed

c) PC unchanged

Instruction 3: step (4)

Memory address	Assembly language	Machine code
0x0000C008	**ADD R3,R1,R2**	**E0 81 30 02**

(4) execute instruction

a) ALU acts and computes addition

b) Contents of R1 and R2 are copied to the input ports of the ALU and the result from ADD is copied into R3

c) Everything happens inside the CPU

d) PC unchanged

Instruction 4: steps (1) and (2)

Memory address	Assembly language	Machine code
0x0000C00C	**STR R3,[R4]**	**E5 84 30 00**

(1) fetch instruction from memory

a) PC is = 0x0000C00C

b) MAR ← PC = 0x0000C00C

c) 0xC00C placed on address bus by MAR

d) CPU issues READ command to memory using a signal on the Control bus

e) Wait for memory to respond

f) 32-bit content at location 0x0000C00C =

=0xE5843000 is returned on data bus to MDR

(2) update current memory address

a) IR ← MDR and 0x E5843000 placed in instruction register (IR)

b) PC is incremented by 4 to become 0x0000C010

Instruction 4: step (3)

Memory address	Assembly language	Machine code
0x0000C00C	**STR R3,[R4]**	**E5 84 30 00**

(3) decode instruction

a) Decoder in control unit recognizes STR instruction

b) Addressing mode is “index” ➔ R4 contains the *address* of the required location in memory (that is, R4 is a “pointer”)

c) The value to be stored/written in memory is contained in R3

Instruction 4: step (4)

Memory address	Assembly language	Machine code
0x0000C00C	**STR R3,[R4]**	**E5 84 30 00**

(4) execute instruction

a) Need to issue a WRITE cycle to place value

b) MAR ← R4 i.e. content of R4 copied to MAR and then placed on address bus by MAR

c) MDR ← R3 i.e. content of R3 copied to MDR and then placed on data bus by MDR

d) CPU issues WRITE command to memory using a signal on the Control bus

e) PC unchanged

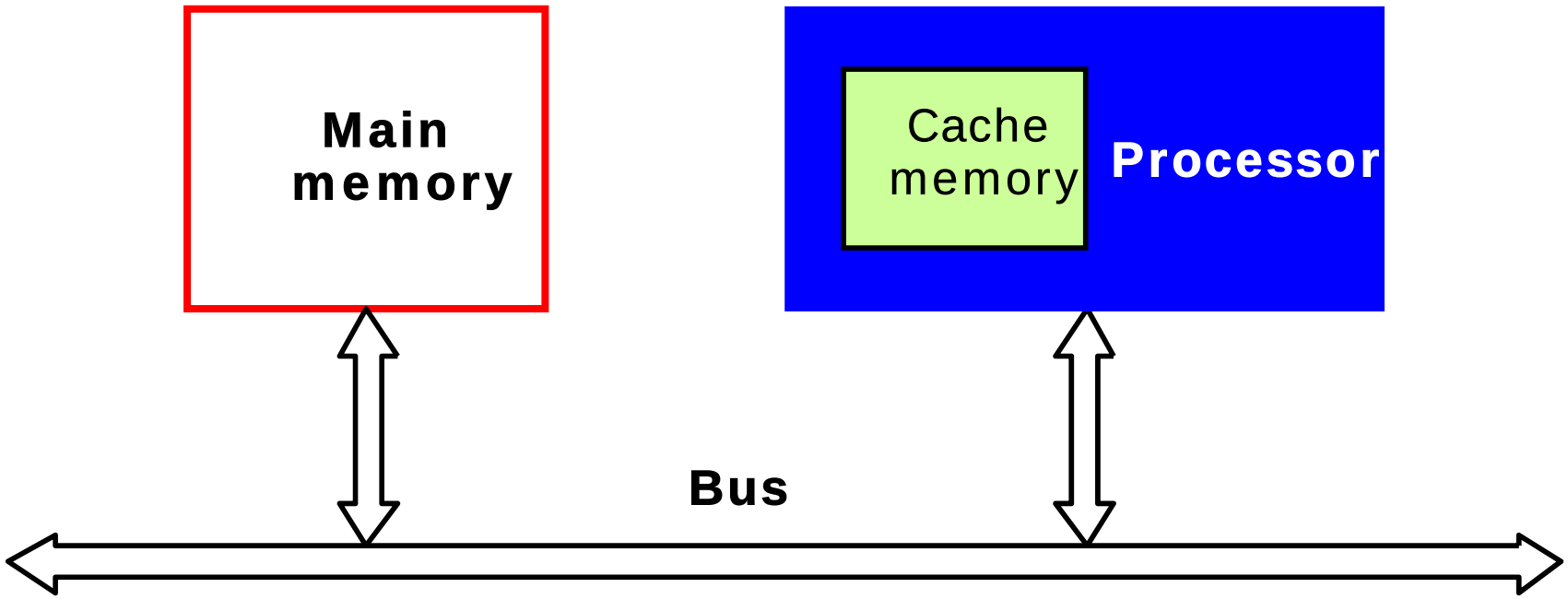
Summary

Memory address	Assembly language	Machine code
0x0000C000	MOV R1,#12	E3 A0 10 0C
0x0000C004	LDR R2,[R5]	E5 95 20 00
0x0000C008	ADD R3,R1,R2	E0 81 30 02
0x0000C00C	STR R3,[R4]	E5 84 30 00

NOTE: All 4 instructions access memory ONCE to fetch the instruction itself

Then, 2 instructions (LDR and STR) access memory again during execution in a similar manner ➔similar ***addressing modes***

Processor Speed versus memory Access Speed

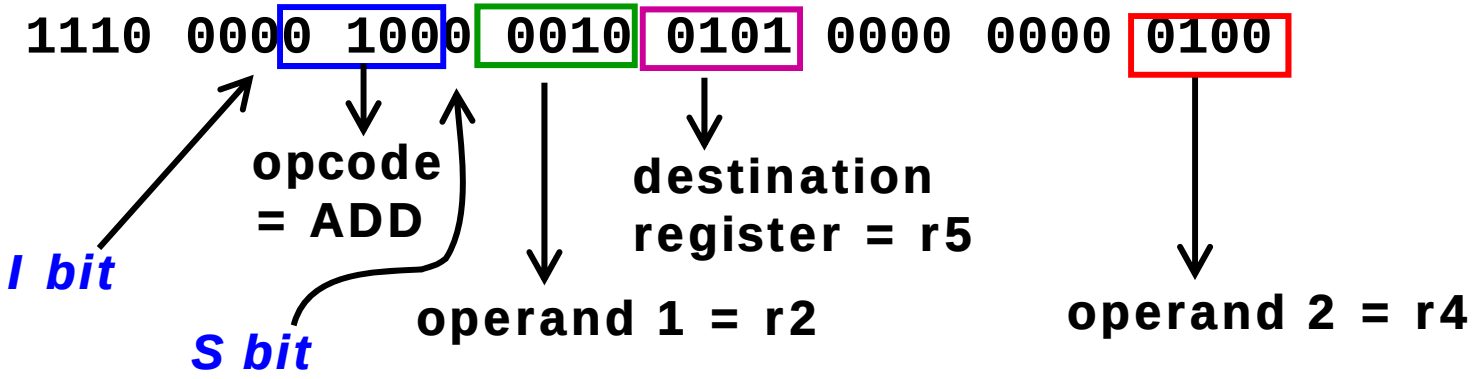


How does the decoder know what instruction it is?

example: fetched instruction = **E0825004**

= **1110 0000 1000 0010 0101 0000 0000 0100**

by reading tables in Appendix B one finds out it is of the kind:



ADD r5,r2,r4

Thus the bytes in the instruction contain information for:

a) type of instruction

b) type of operands to expect (*ADDRESSING MODES*)