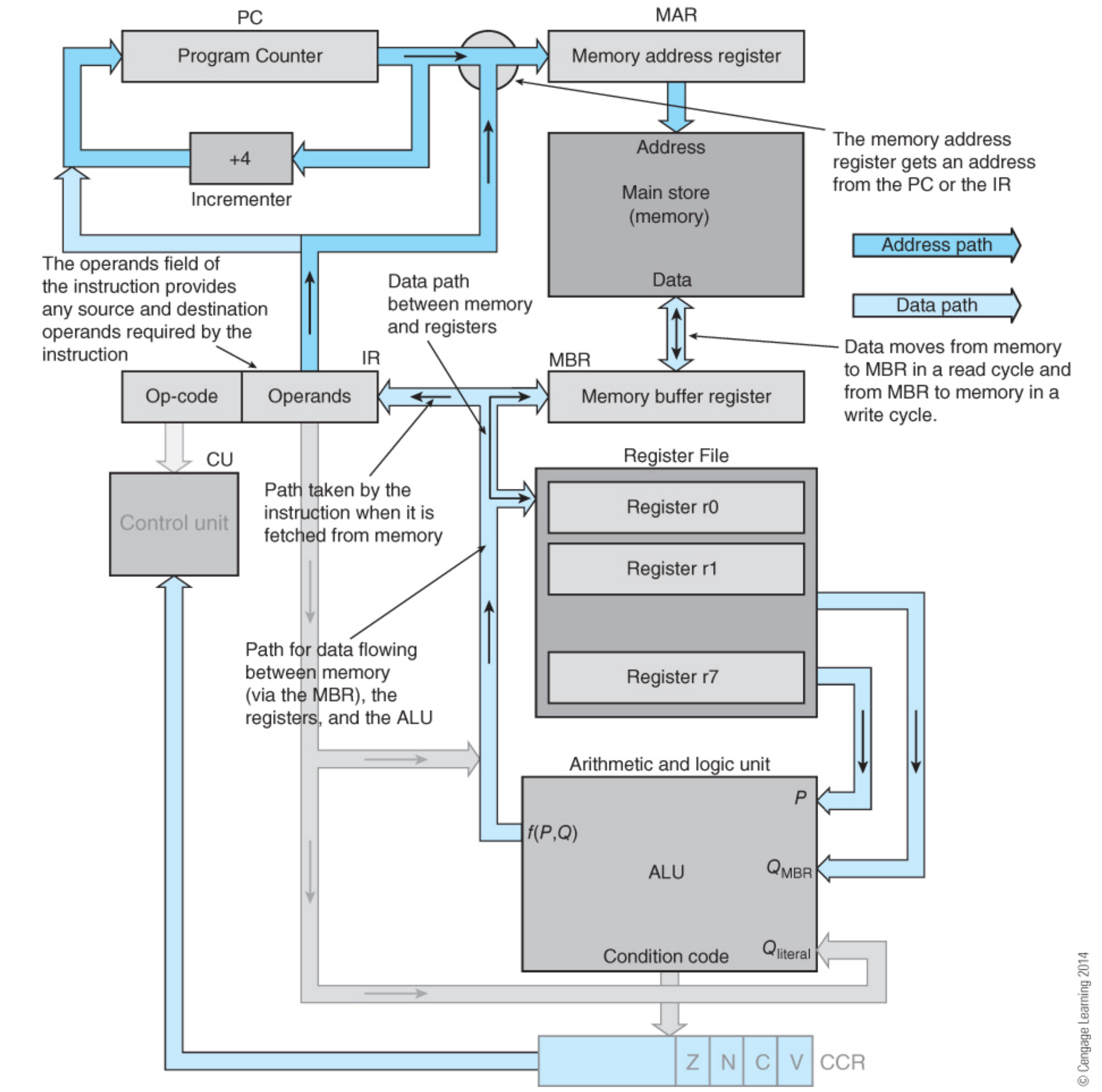
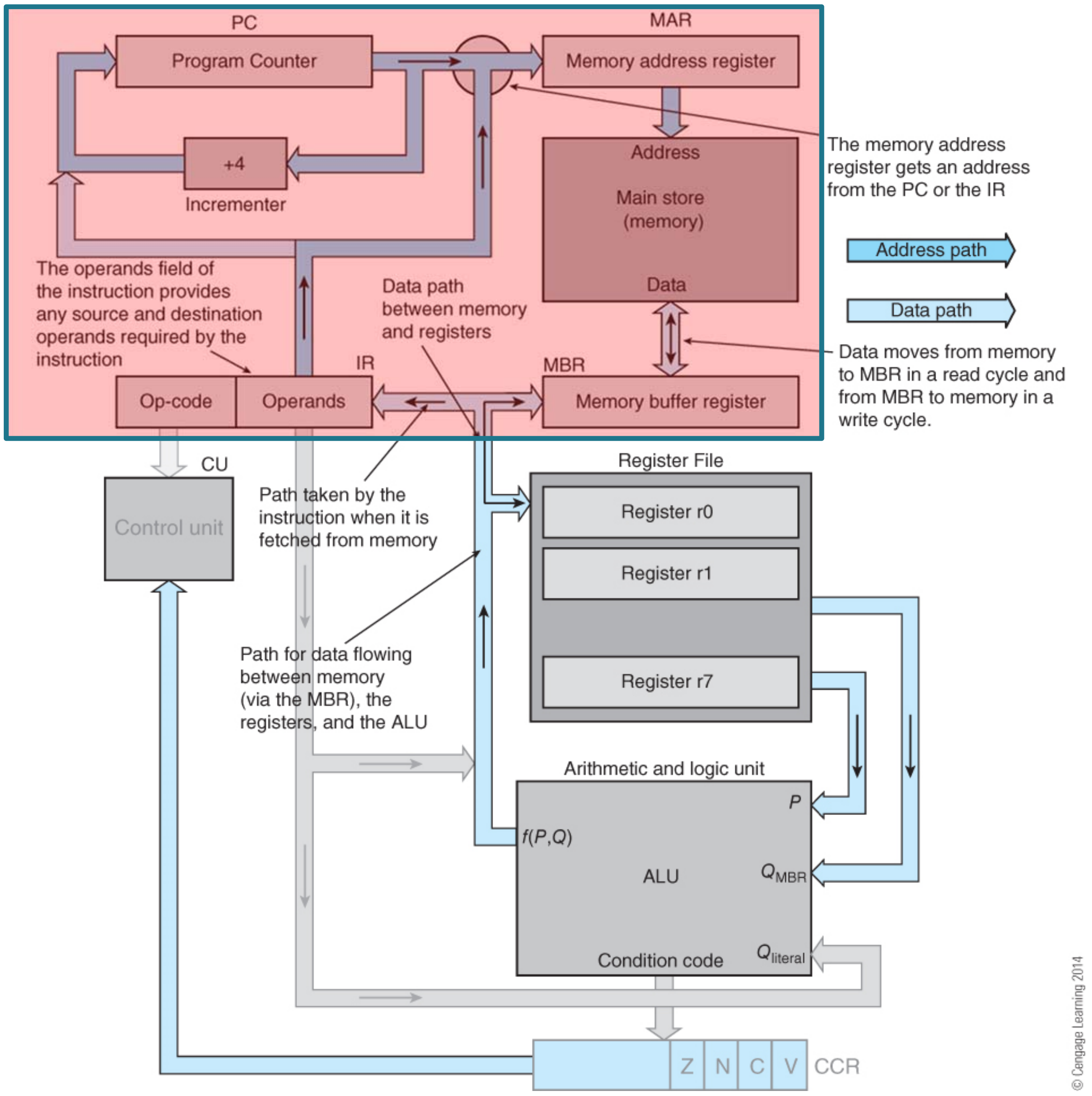
FIGURE 3.2 Partial structure of a hypothetical stored program machine


PC
Program Counter
MAR
Memory address register
+4
Incrementer
Address
Main store (memory)
Data
The memory address register gets an address from the PC or the IR
Address path
Data path
The operands field of the instruction provides any source and destination operands required by the instruction
Data path between memory and registers
IR
Op-code
Operands
MBR
Memory buffer register
Data moves from memory to MBR in a read cycle and from MBR to memory in a write cycle.
CU
Control unit
Path taken by the instruction when it is fetched from memory
Register File
Register r0
Register r1
Register r7
Path for data flowing between memory (via the MBR), the registers, and the ALU
Arithmetic and logic unit
P
f(P,Q)
ALU
Q_MBR
Condition code
Q_literal
Z N C V CCR
© Cengage Learning 2014

Structure of a Computer

- We are going to use the ARM processor to introduce assembly language and a modern ISA.
- *However*, we begin with the description of a very simple *hypothetical* computer to keep things simple.

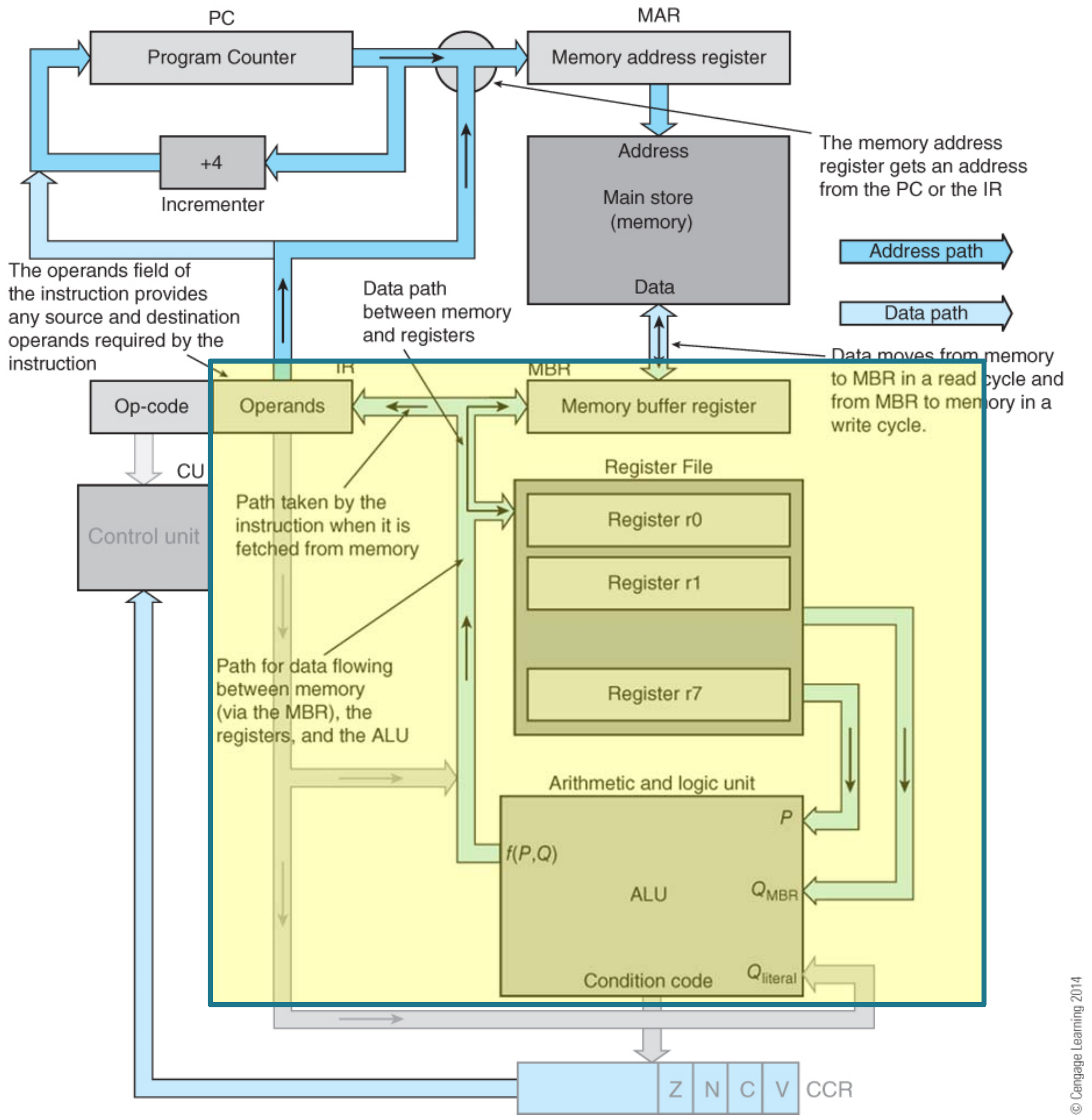
FIGURE 3.2 Partial structure of a hypothetical stored program machine


PC
Program Counter
MAR
Memory address register
+4
Incrementer
Address
Main store (memory)
Data
The memory address register gets an address from the PC or the IR
Address path
Data path
The operands field of the instruction provides any source and destination operands required by the instruction
Data path between memory and registers
IR
Op-code
Operands
MBR
Memory buffer register
Data moves from memory to MBR in a read cycle and from MBR to memory in a write cycle.
CU
Control unit
Path taken by the instruction when it is fetched from memory
Register File
Register r0
Register r1
Register r7
Path for data flowing between memory (via the MBR), the registers, and the ALU
Arithmetic and logic unit
P
f(P,Q)
ALU
Q_MBR
Condition code
Q_literal
Z N C V CCR
© Cengage Learning 2014

Structure of a Computer

- In the ***fetch phase***, the program counter supplies the address of the next instruction to be executed to the MAR to read this instruction ***and*** the PC is incremented by the size of an instruction.
- The instruction is read and loaded into the memory buffer register, MBR, ***and*** then copied to the instruction register, IR where the op-code is decoded.

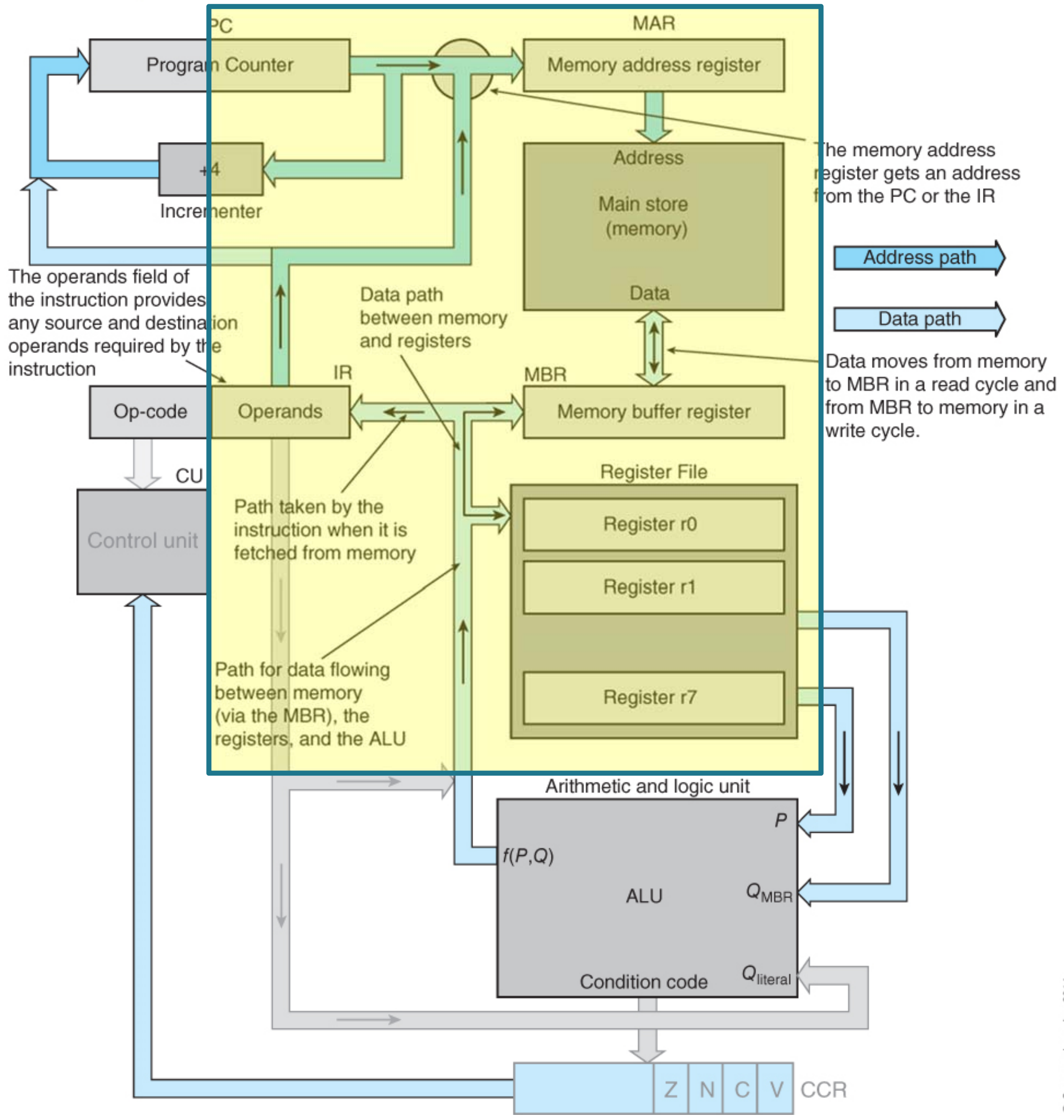
FIGURE 3.2 Partial structure of a hypothetical stored program machine


PC
Program Counter
MAR
Memory address register
+4
Incrementer
Address
Main store (memory)
Data
The memory address register gets an address from the PC or the IR
Address path
Data path
The operands field of the instruction provides any source and destination operands required by the instruction
Data path between memory and registers
Data moves from memory to MBR in a read cycle and from MBR to memory in a write cycle.
IR
MBR
Op-code
Operands
Memory buffer register
CU
Control unit
Register File
Register r0
Register r1
Register r7
Path taken by the instruction when it is fetched from memory
Path for data flowing between memory (via the MBR), the registers, and the ALU
Arithmetic and logic unit
P
f(P,Q)
ALU
Q_MBR
Condition code
Q_literal
Z N C V CCR
© Cengage Learning 2014

Structure of a Computer

- In the ***execute phase***, the operands may be read from the *register file*, transferred to the ALU (*arithmetic and logic unit*) where they are operated on and then the result passed to the *destination register*.
  This is what we called, *register-to-register* operation

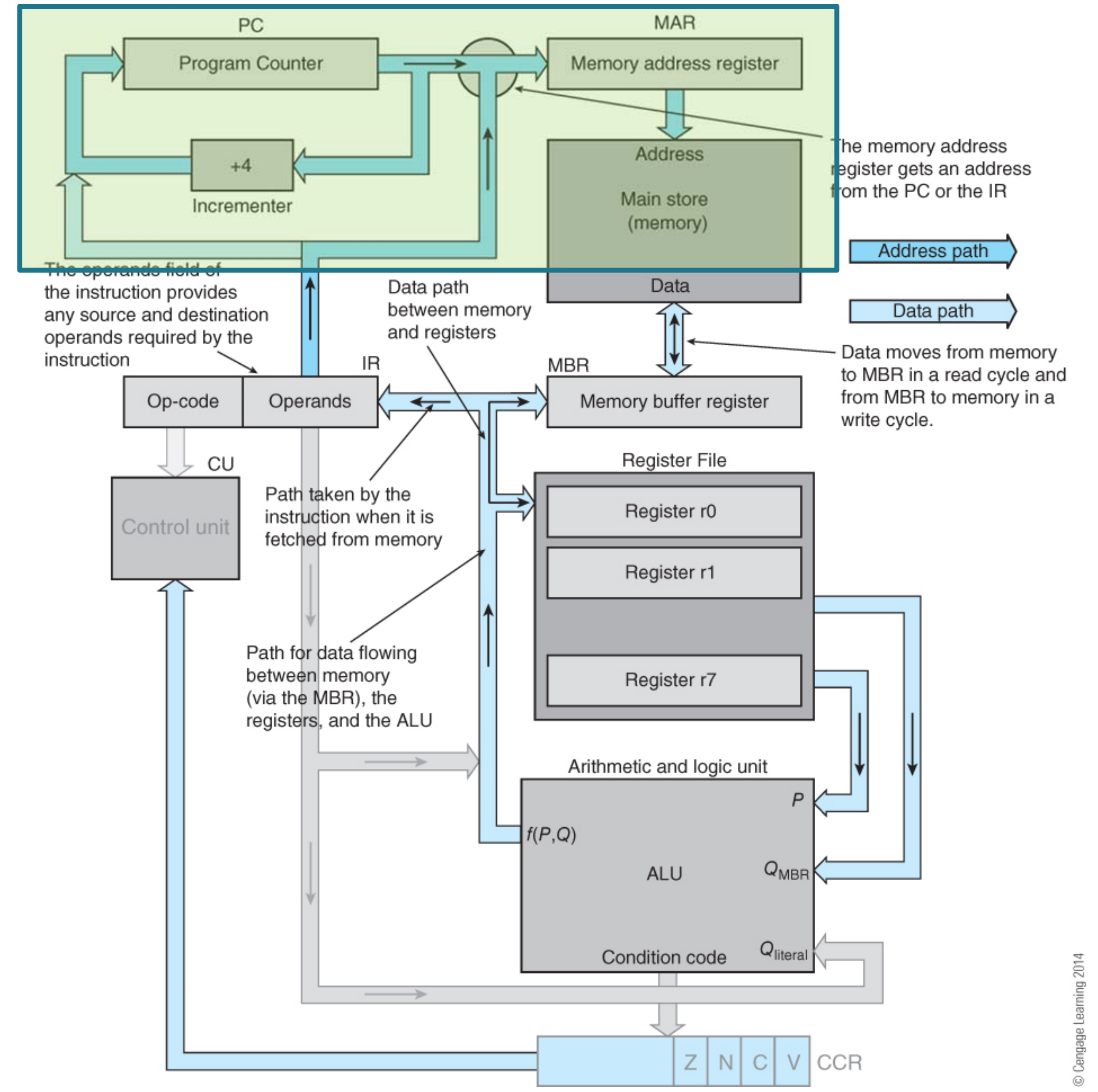
FIGURE 3.2 Partial structure of a hypothetical stored program machine


PC
MAR
Program Counter
Memory address register
+4
Incrementer
Address
Main store (memory)
Data
The memory address register gets an address from the PC or the IR
Address path
Data path
The operands field of the instruction provides any source and destination operands required by the instruction
Data path between memory and registers
IR
MBR
Op-code
Operands
Memory buffer register
Data moves from memory to MBR in a read cycle and from MBR to memory in a write cycle.
CU
Control unit
Register File
Register r0
Register r1
Register r7
Path taken by the instruction when it is fetched from memory
Path for data flowing between memory (via the MBR), the registers, and the ALU
Arithmetic and logic unit
P
f(P,Q)
ALU
Q_MBR
Condition code
Q_literal
Z N C V CCR
© Cengage Learning 2014

Structure of a Computer

- If the operation requires a memory access (e.g., a load or store), the memory address in the instruction register is sent to the MAR and a read or write operation performed.

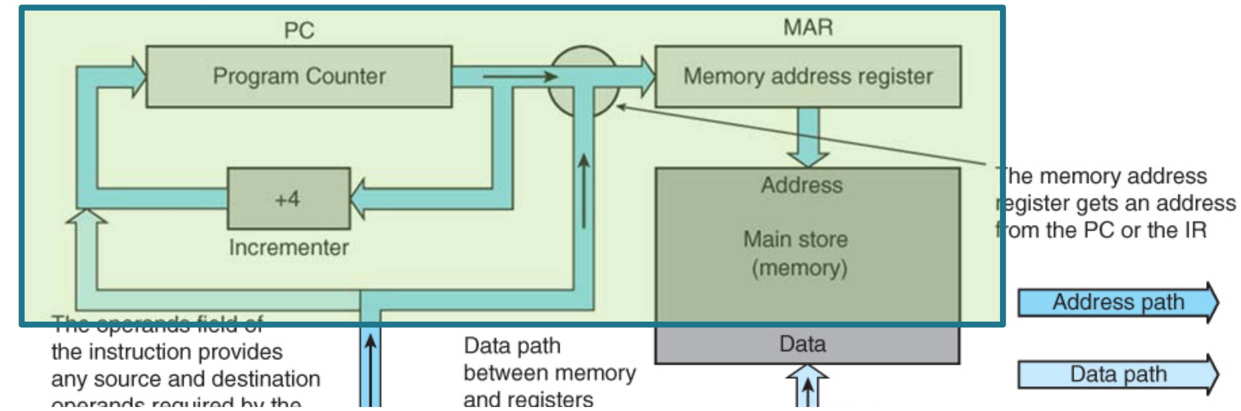
FIGURE 3.2 Partial structure of a hypothetical stored program machine


PC
Program Counter
MAR
Memory address register
+4
Incrementer
Address
Main store
(memory)
Data
The memory address register gets an address from the PC or the IR
Address path
Data path
The operands field of the instruction provides any source and destination operands required by the instruction
Data path between memory and registers
IR
MBR
Op-code
Operands
Memory buffer register
Data moves from memory to MBR in a read cycle and from MBR to memory in a write cycle.
CU
Control unit
Path taken by the instruction when it is fetched from memory
Register File
Register r0
Register r1
Register r7
Path for data flowing between memory (via the MBR), the registers, and the ALU
Arithmetic and logic unit
P
f(P,Q)
ALU
QMBR
Condition code
Qliteral
Z
N
C
V
CCR
© Cengage Learning 2014

Structure of a Computer

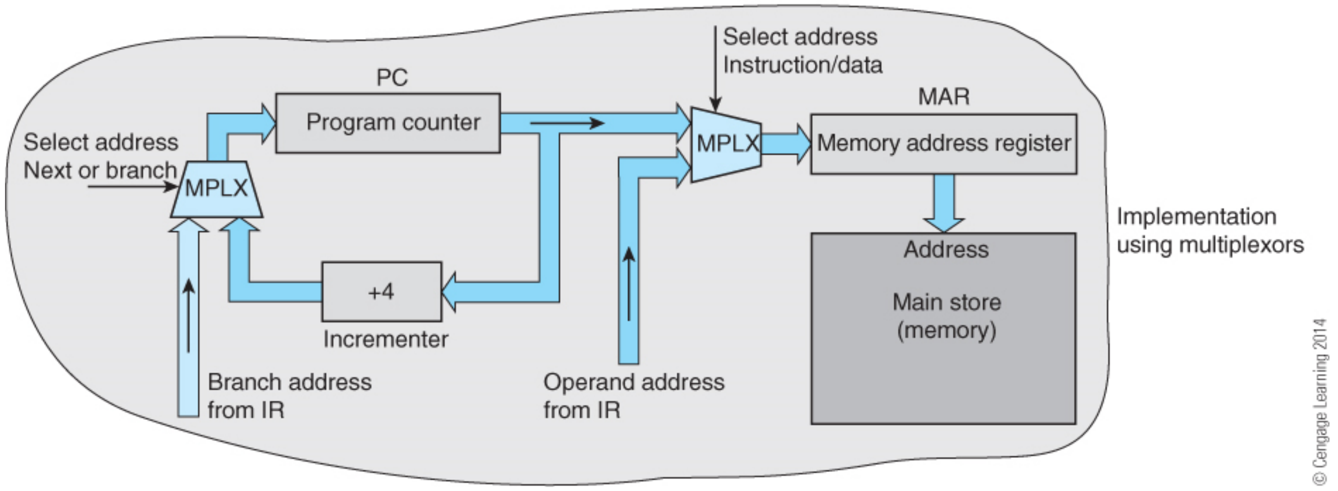
- But, how can we combine two input data lines together?

FIGURE 3.2 Partial structure of a hypothetical stored program machine


PC
Program Counter
MAR
Memory address register
+4
Incrementer
Address
Main store
(memory)
Data
The memory address register gets an address from the PC or the IR
Address path
Data path
The operands field of the instruction provides any source and destination operands required by the
Data path between memory and registers

Structure of a Computer

- But, how can we combine two input data lines together?



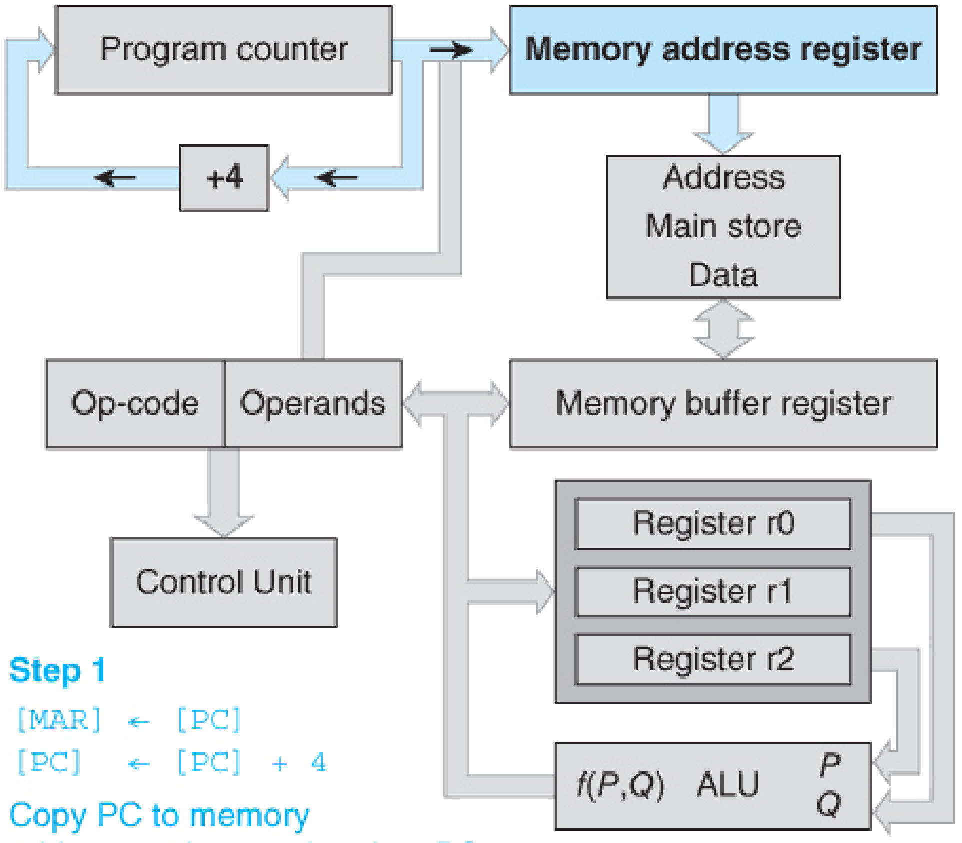
Structure of a Computer

- Fetch/execute cycle in RTL

```
FETCH   [MAR] ← [PC]         ;copy PC to MAR
        [PC] ← [PC] + 4      ;increment PC
        [MBR]  ← [[MAR]]     ;read instruction pointed at by MAR
        [IR] ← [MBR]         ;copy instruction in  MBR to  IR

LDR     [MAR] ← [IR(address)]  ;copy operand address from IR to MAR
        [MBR] ← [[MAR]]      ;read operand value from memory
        [r1] ← [MBR]         ;add the operand to register r1
```

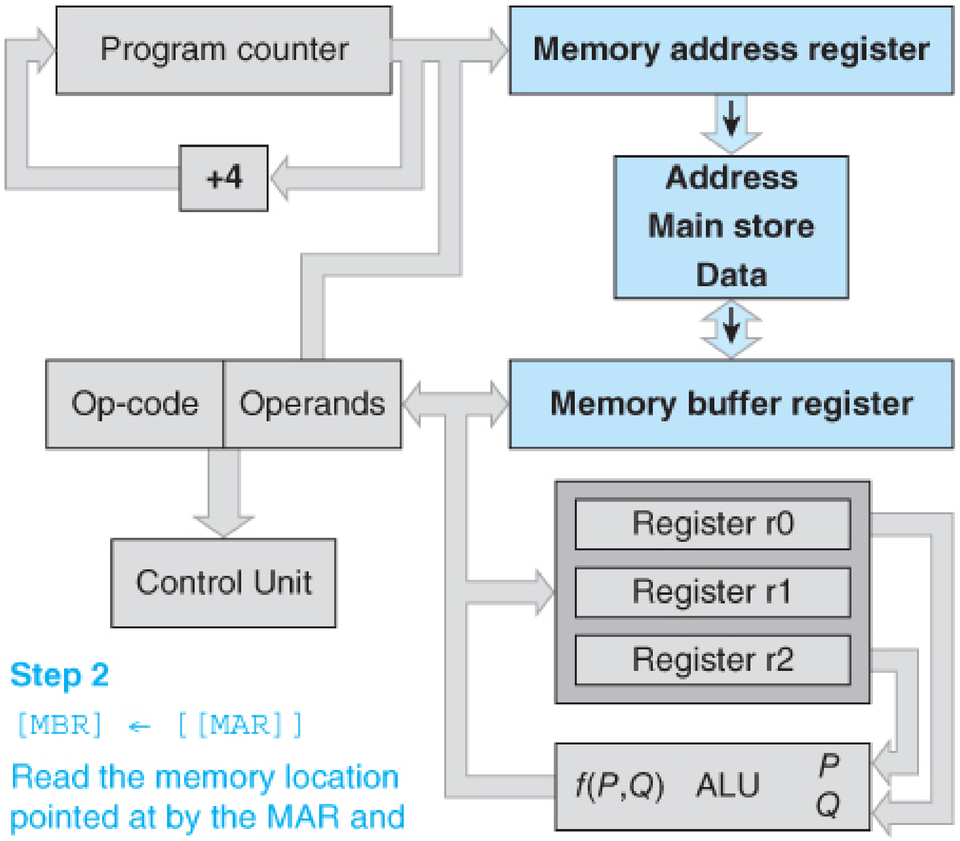
Fetching and Executing an Instruction



Step 1

```
[MAR] ← [PC]
[PC]  ← [PC] + 4
```

Copy PC to memory address register and update PC.

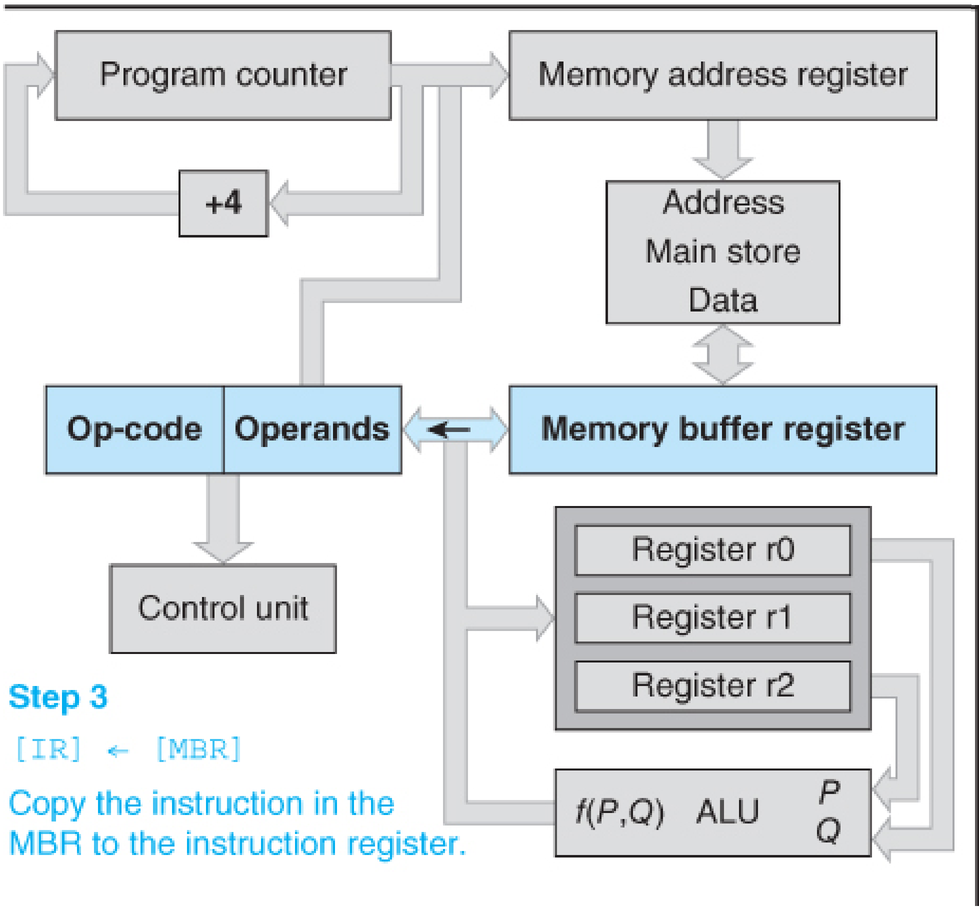


Step 2

```
[MBR] ← [[MAR]]
```

Read the memory location pointed at by the MAR and put the instruction in the MBR.

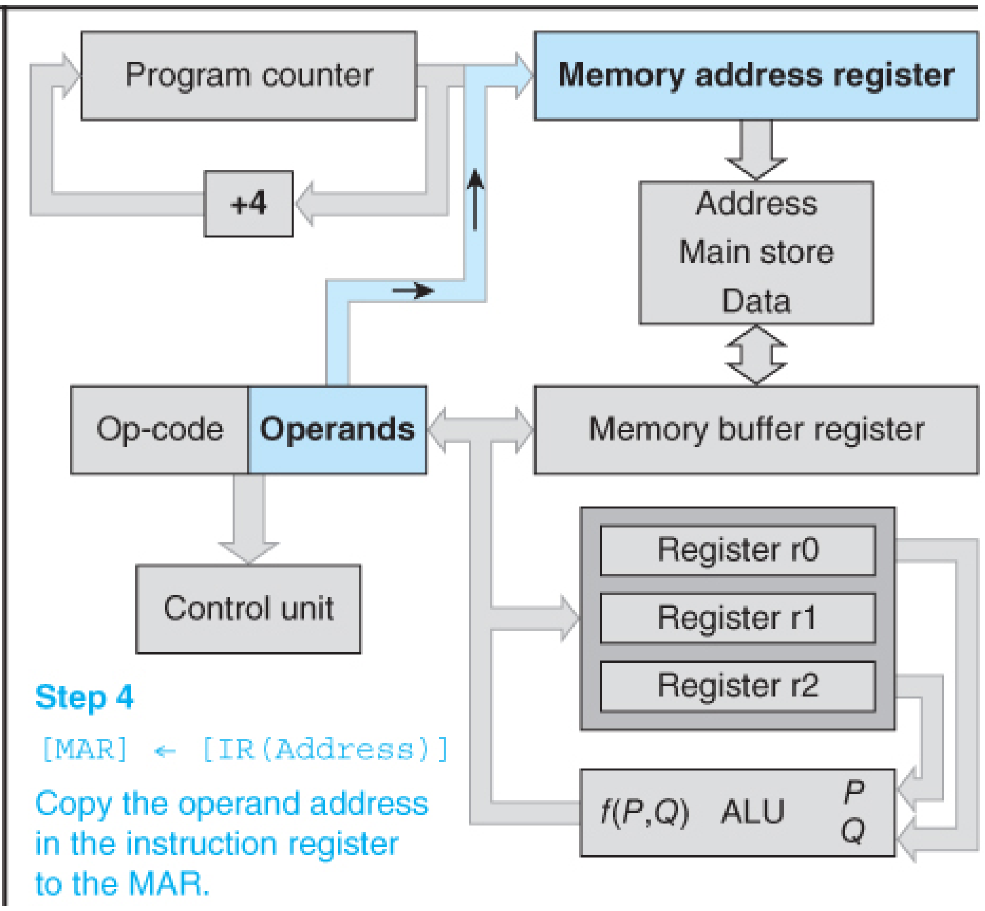
Fetching and Executing an Instruction



Step 3

`[IR] ← [MBR]`

Copy the instruction in the MBR to the instruction register.

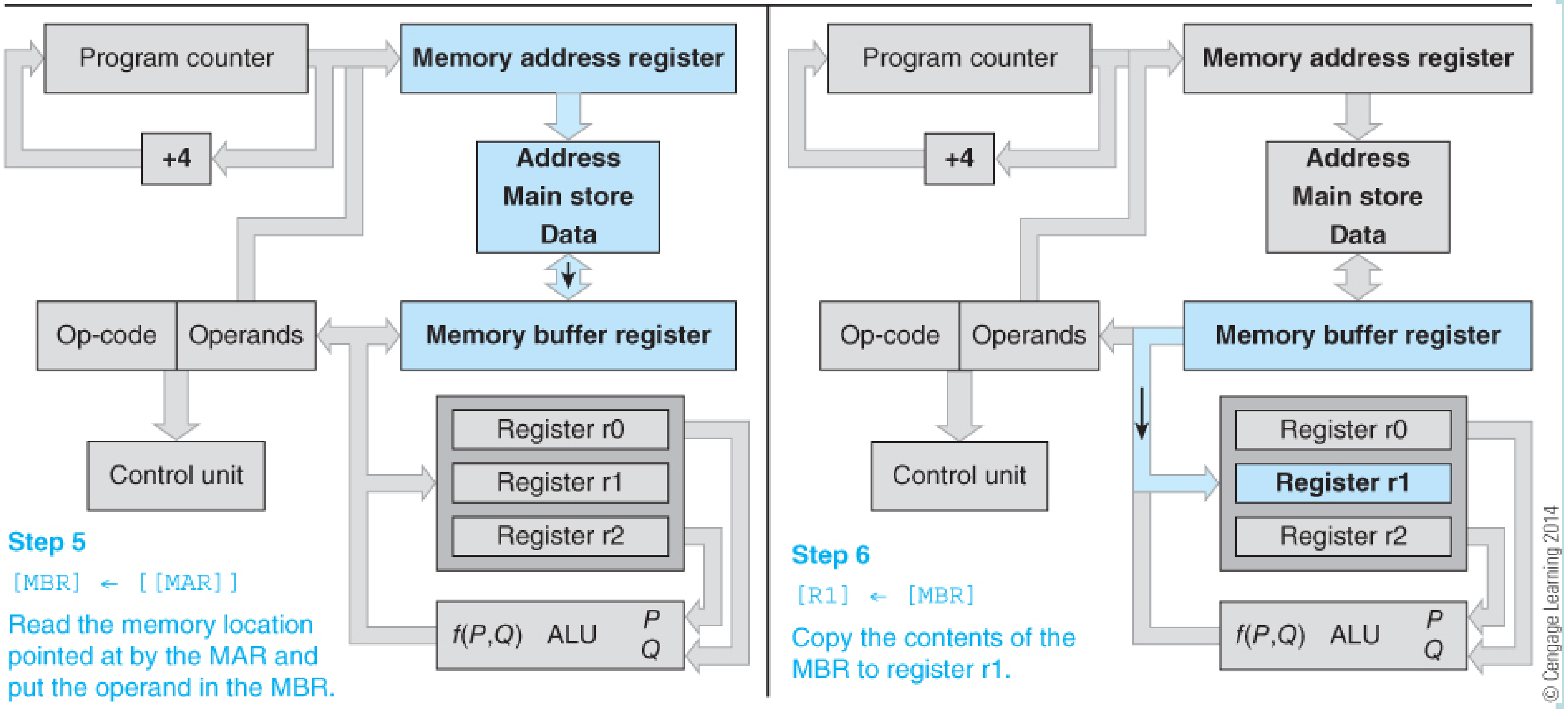


Step 4

`[MAR] ← [IR(Address)]`

Copy the operand address in the instruction register to the MAR.

Fetching and Executing an Instruction



Step 5

`[MBR] ← [[MAR]]`

Read the memory location pointed at by the MAR and put the operand in the MBR.

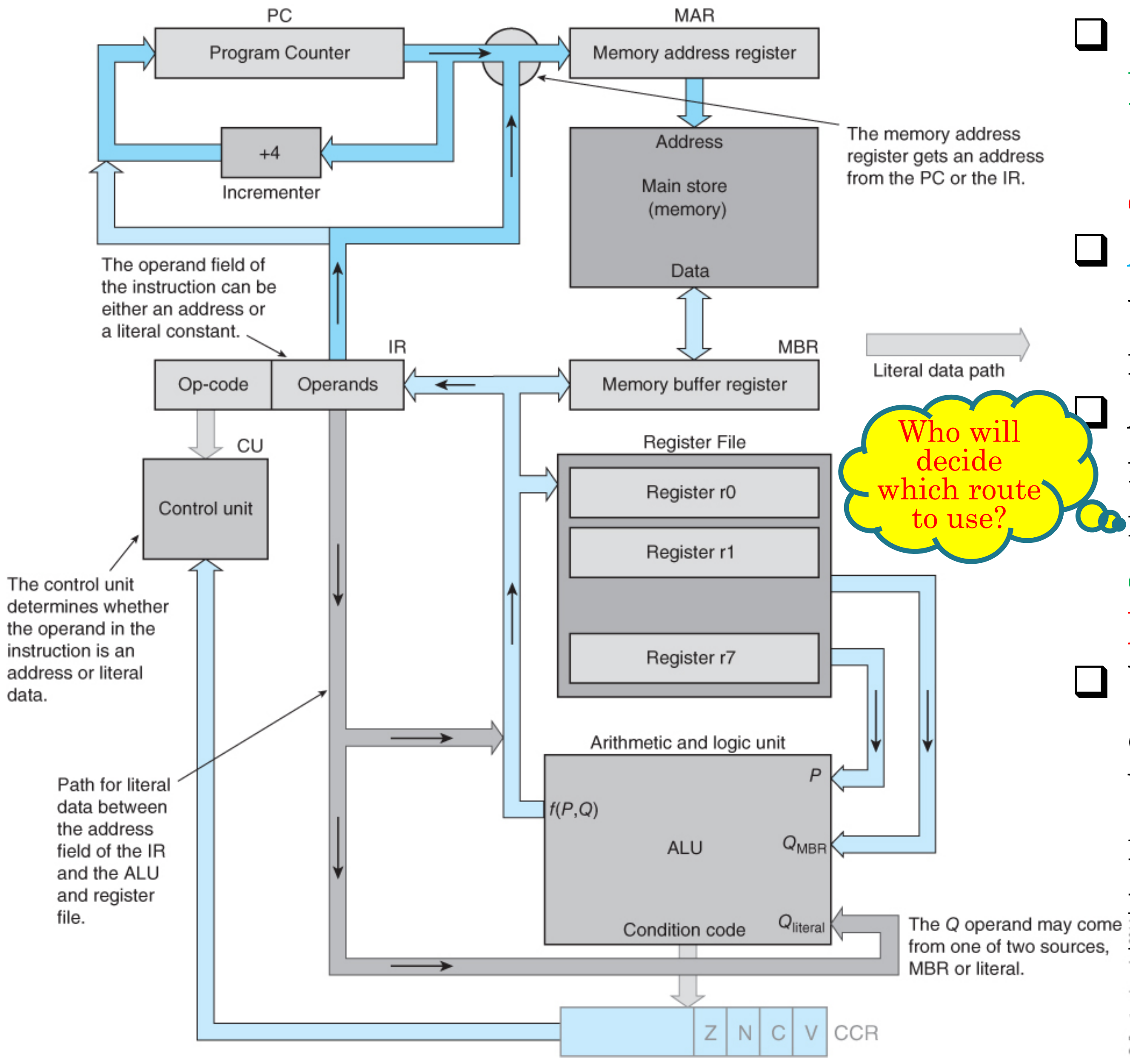
Step 6

`[R1] ← [MBR]`

Copy the contents of the MBR to register r1.

Dealing with Constants

FIGURE 3.4 Information paths for literal operands


PC
Program Counter
MAR
Memory address register
+4
Incrementer
Address
Main store (memory)
Data
The memory address register gets an address from the PC or the IR.
The operand field of the instruction can be either an address or a literal constant.
IR
Op-code
Operands
MBR
Memory buffer register
Literal data path
CU
Control unit
Register File
Register r0
Register r1
Register r7
The control unit determines whether the operand in the instruction is an address or literal data.
Arithmetic and logic unit
P
f(P,Q)
ALU
Q_MBR
Condition code
Q_literal
The Q operand may come from one of two sources, MBR or literal.
Path for literal data between the address field of the IR and the ALU and register file.
Z N C V CCR
© Cengage Learning 2014
Who will decide which route to use?

- Suppose we want to load the *number 1234 itself* (*a.k.a. literal operand*) into register r1.
- ADD **r0**,r1,#25 adds the value 25 to the content of r1 and puts the sum in r0
- A path from the instruction register, IR, routes a literal operand to *either* the register file, MBR, and ALU
- When ADD **r0**,r1,#25 is executed, the operand to be added to r1, #25, is routed from the operand field of the IR, rather than from the memory system via the MBR.