Microcomputer Organization

II CPU

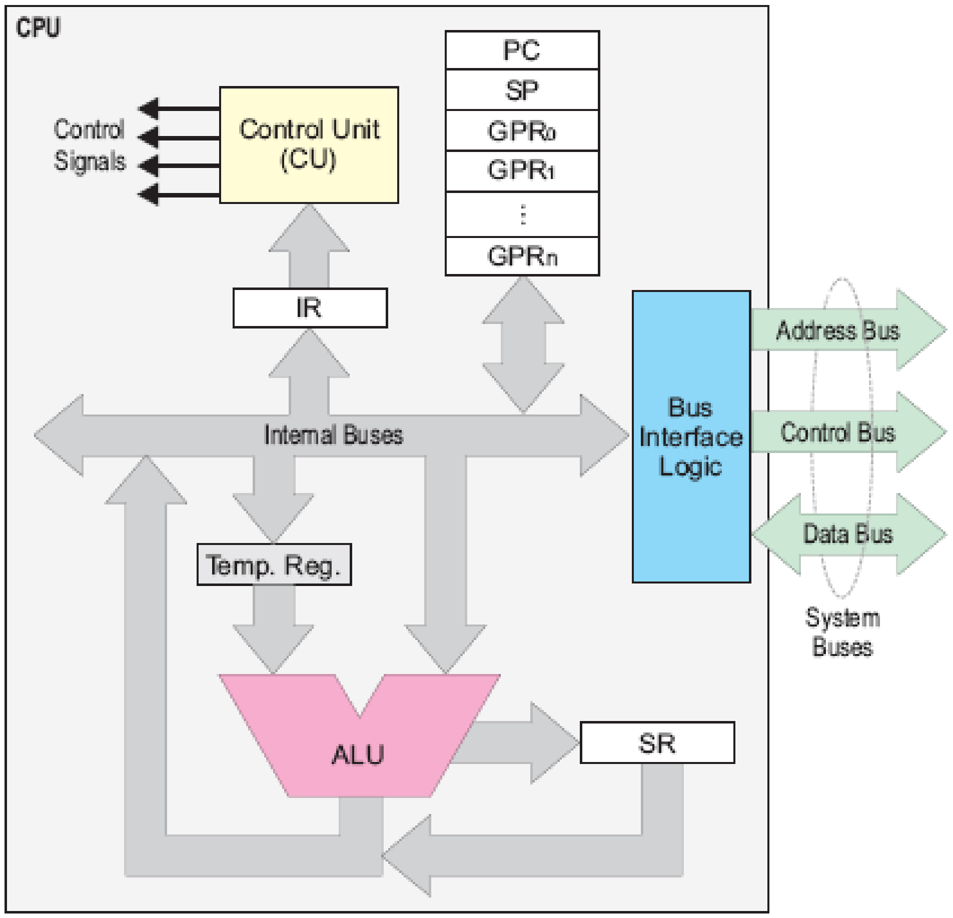
CPU Components

- Hardware components
  - Control Unit (CU)
  - Registers
  - Arithmetic Logic Unit (ALU)
  - Bus Interface Logic unit (BIL)
- Software Components
  - Instruction Set
  - Addressing modes

CPU

Data Path and Control Path

- Data Path: HW components that perform operations
  - ALU
  - Registers and Internal Buses
  - Specialized units
- Control Path: HW components controlling system operation
  - CU
  - BIL
  - Timing and synchronization units

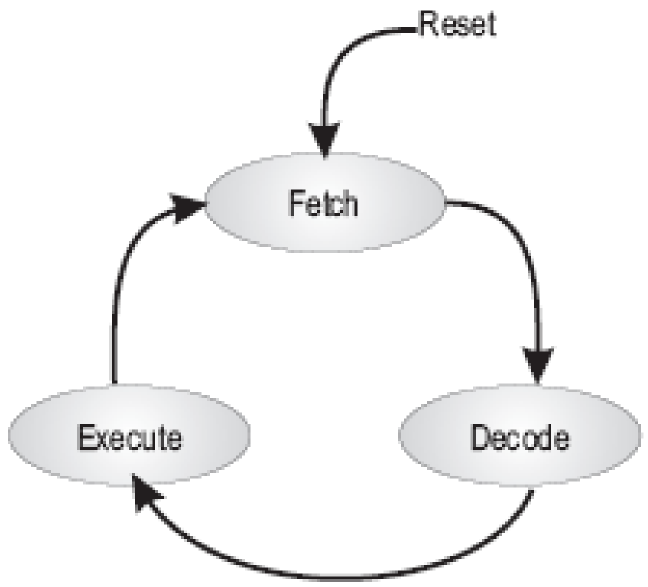


CPU: Registers

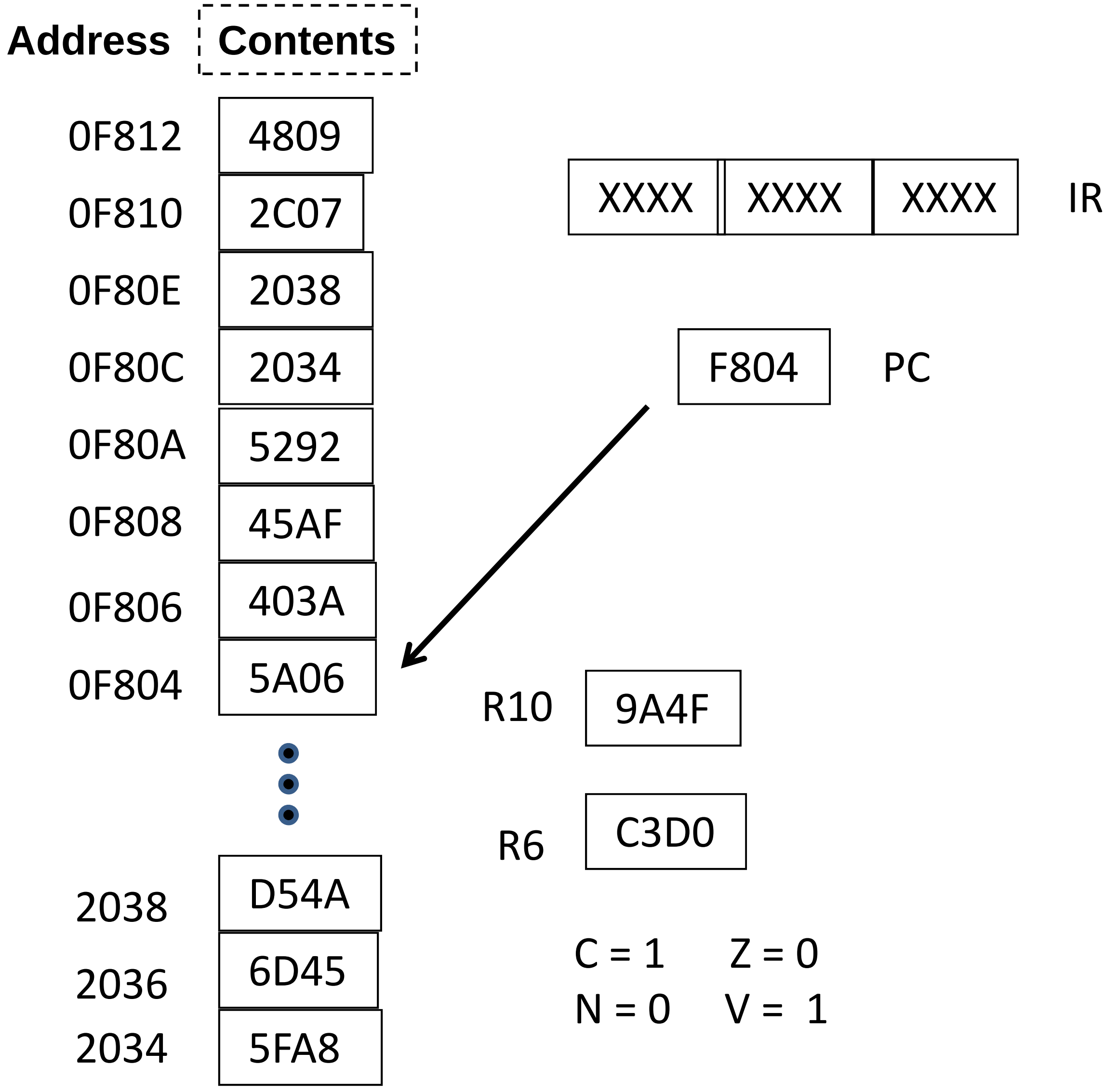
- Special Purpose Registers
  - Instruction Register (IR)
  - Program Counter (PC)
  - Stack Pointer (SP)
  - Status Register (SR)
- General Purpose registers
- “Invisible registers” not available to programmer

Instruction Cycle or CPU Cycle: Fetch-Decode Execute

- **Fetch: The CU brings a new instruction from memory through BIL**
  - Register PC provides the address of instruction to be fetched
  - Instruction is stored in IR
- **Decode: instruction meaning is deciphered**
- **Execute: CU commands the corresponding units to perform the actions.**
  - PC has new address after execution
- **Reset: A defined state after power up or after a reset occurs**



Example: Initial State



Address	Contents
0F812	4809
0F810	2C07
0F80E	2038
0F80C	2034
0F80A	5292
0F808	45AF
0F806	403A
0F804	5A06
⋮	⋮
2038	D54A
2036	6D45
2034	5FA8

IR	XXXX XXXX XXXX
PC	F804
R10	9A4F
R6	C3D0

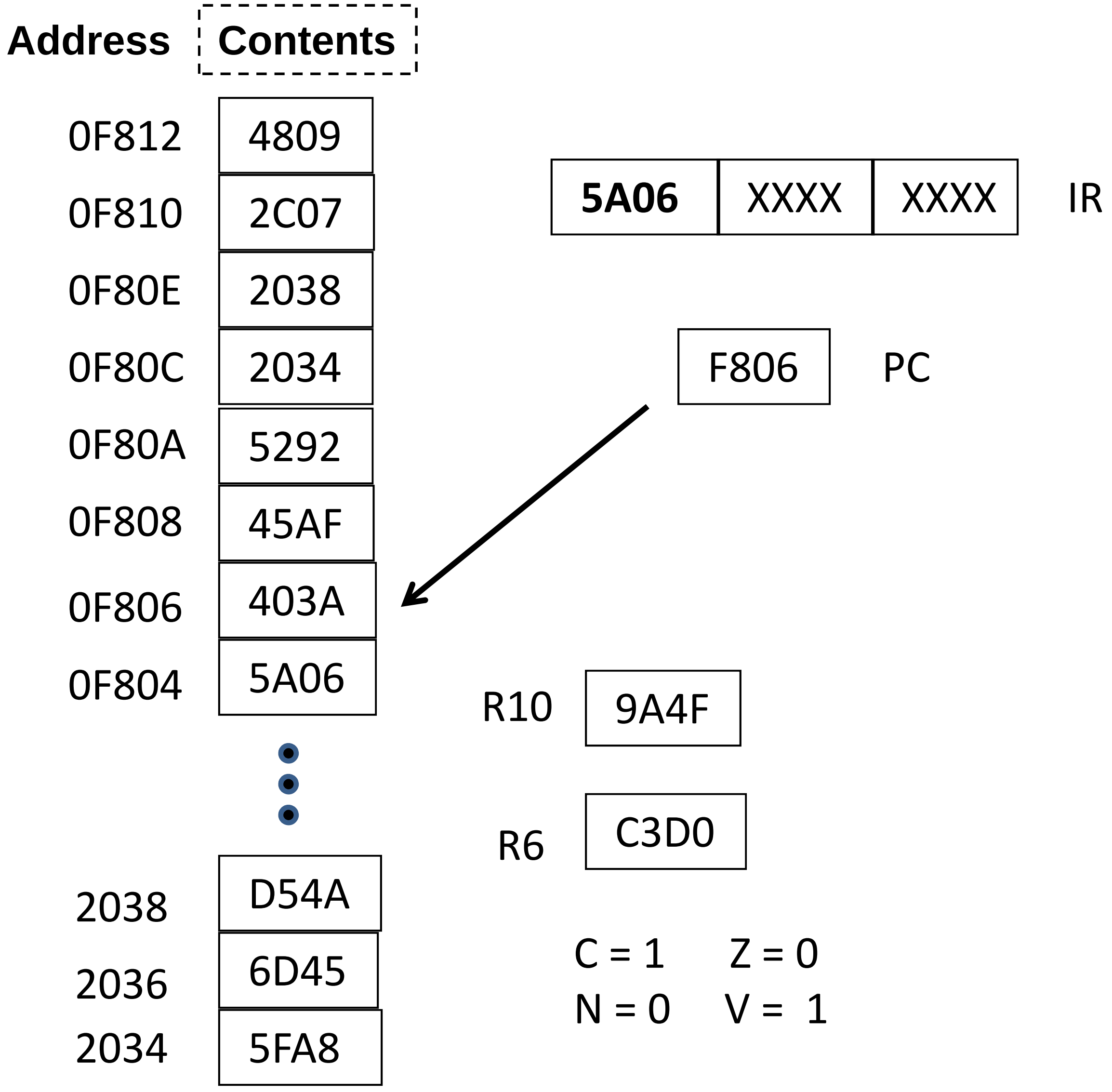
C = 1 Z = 0
N = 0 V = 1

Just after a previous cycle:

a) Contents of IR is irrelevant.

b) PC points to next instruction

FETCH

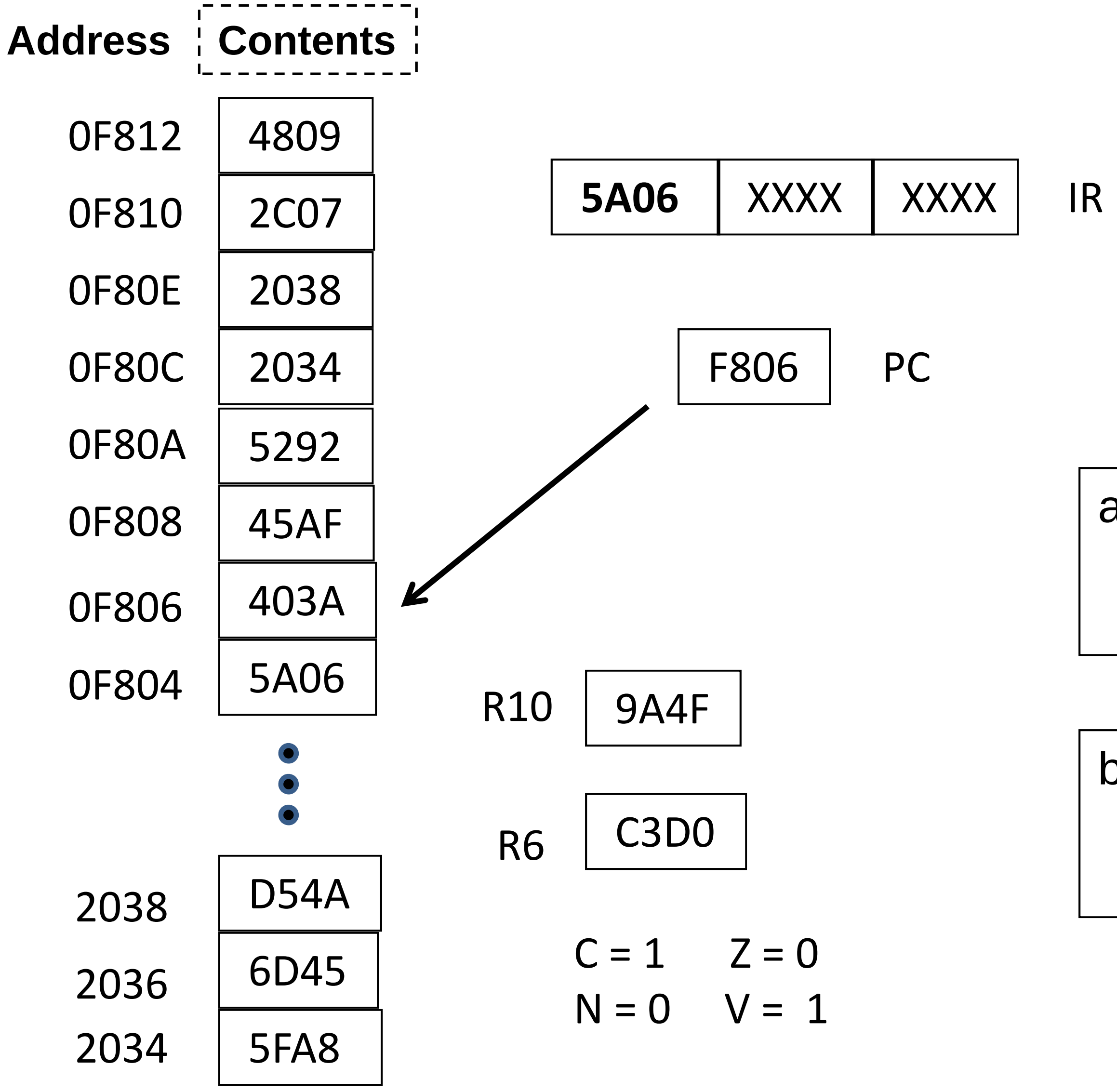


CU puts PC contents in Address Bus (using BIL unit) and reads (using Data Bus and Control Bus) memory contents and puts result into Instruction Register IR

PC increases its value pointing to next address.

(Fetched word in this first movement is the **Instruction Word)**

DECODE



Address	Contents
0F812	4809
0F810	2C07
0F80E	2038
0F80C	2034
0F80A	5292
0F808	45AF
0F806	403A
0F804	5A06
⋮	⋮
2038	D54A
2036	6D45
2034	5FA8

5A06	XXXX	XXXX	IR

PC	F806
R10	9A4F
R6	C3D0

C = 1 Z = 0
N = 0 V = 1

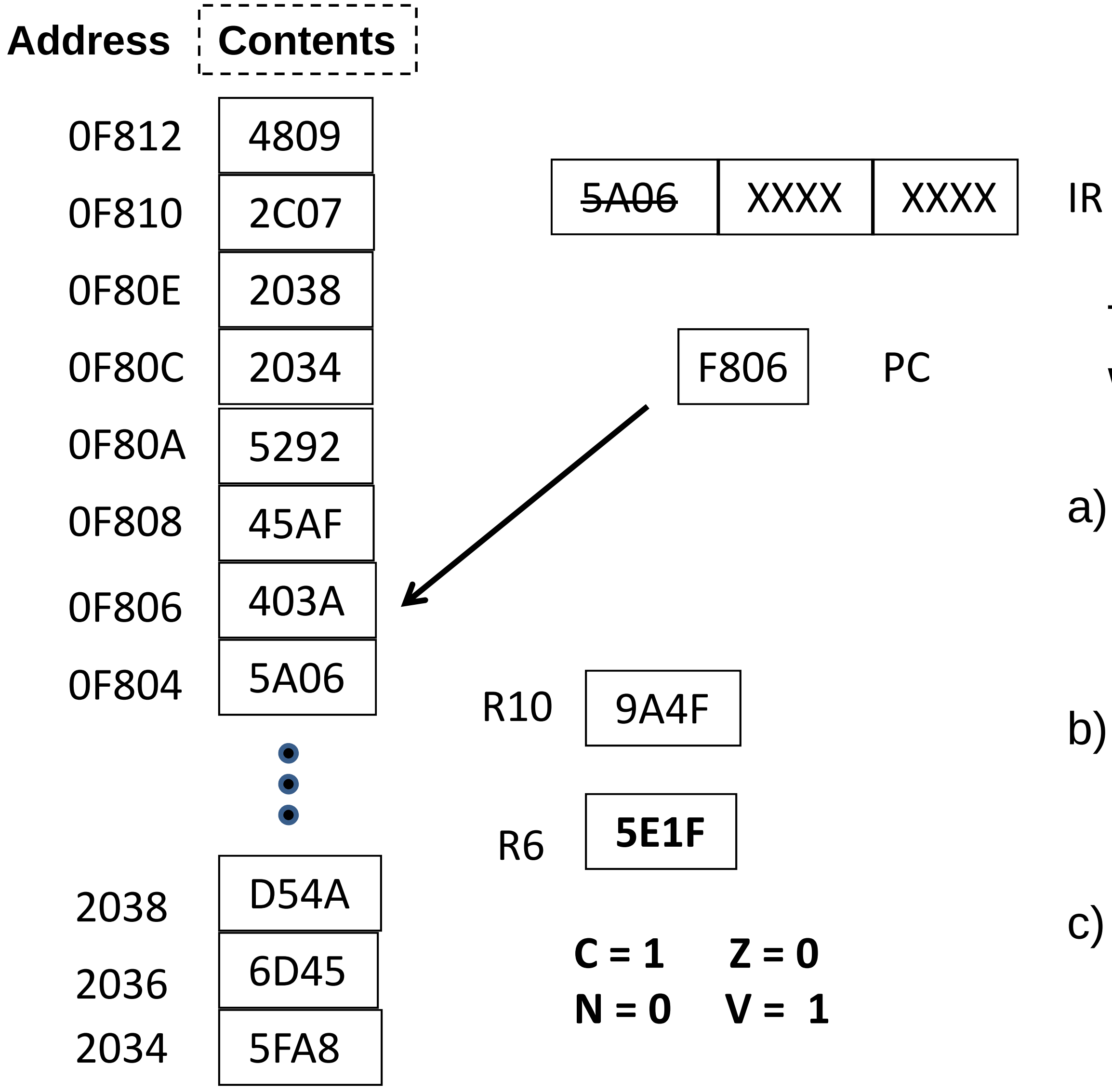
CU decodes:
Add contents of R10 to contents of R6

The information is complete. so decoding is finished.

a) In Register Transfer Notation (RTN):
R6 ← R6 + R10

b) Decoding is finished and PC is pointing to address following this instruction

EXECUTE



The processor executes what decoding indicated:

a) Old contents of destination is lost and has been replaced with new result

b) Flags have been affected by this instruction

c) IR contents is the same, but it is irrelevant

FETCH

Address	**Contents**
0F812	4809
0F810	2C07
0F80E	2038
0F80C	2034
0F80A	5292
0F808	45AF
0F806	403A
0F804	5A06
⋮	⋮
2038	D54A
2036	6D45
2034	5FA8

403A	XXXX	XXXX	IR

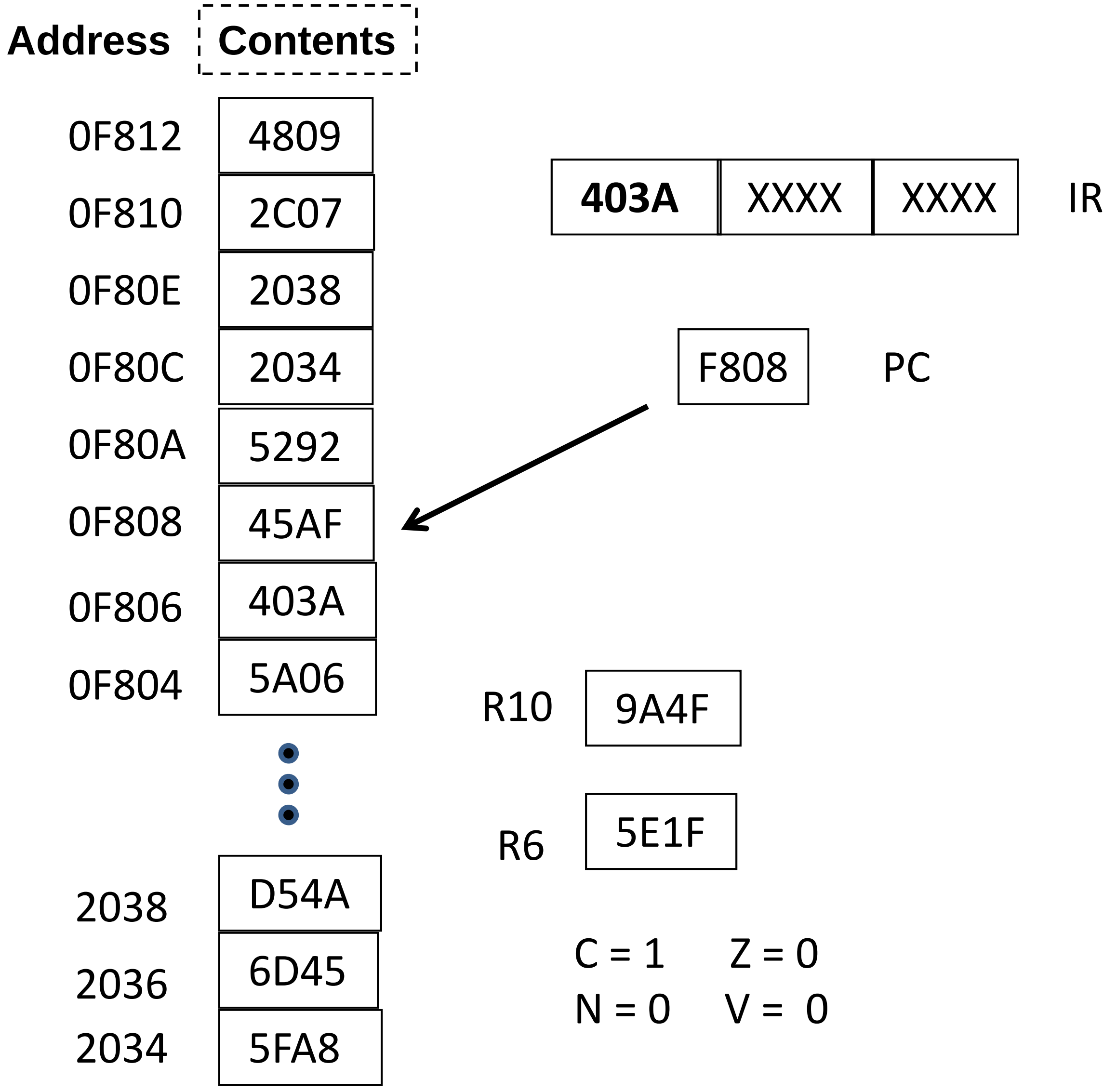
F808	PC

R10	9A4F
R6	5E1F

C = 1   Z = 0
N = 0   V = 0

CU fetches <u>Instruction word</u> and increments PC which is now pointing to next address.

DECODE (1)

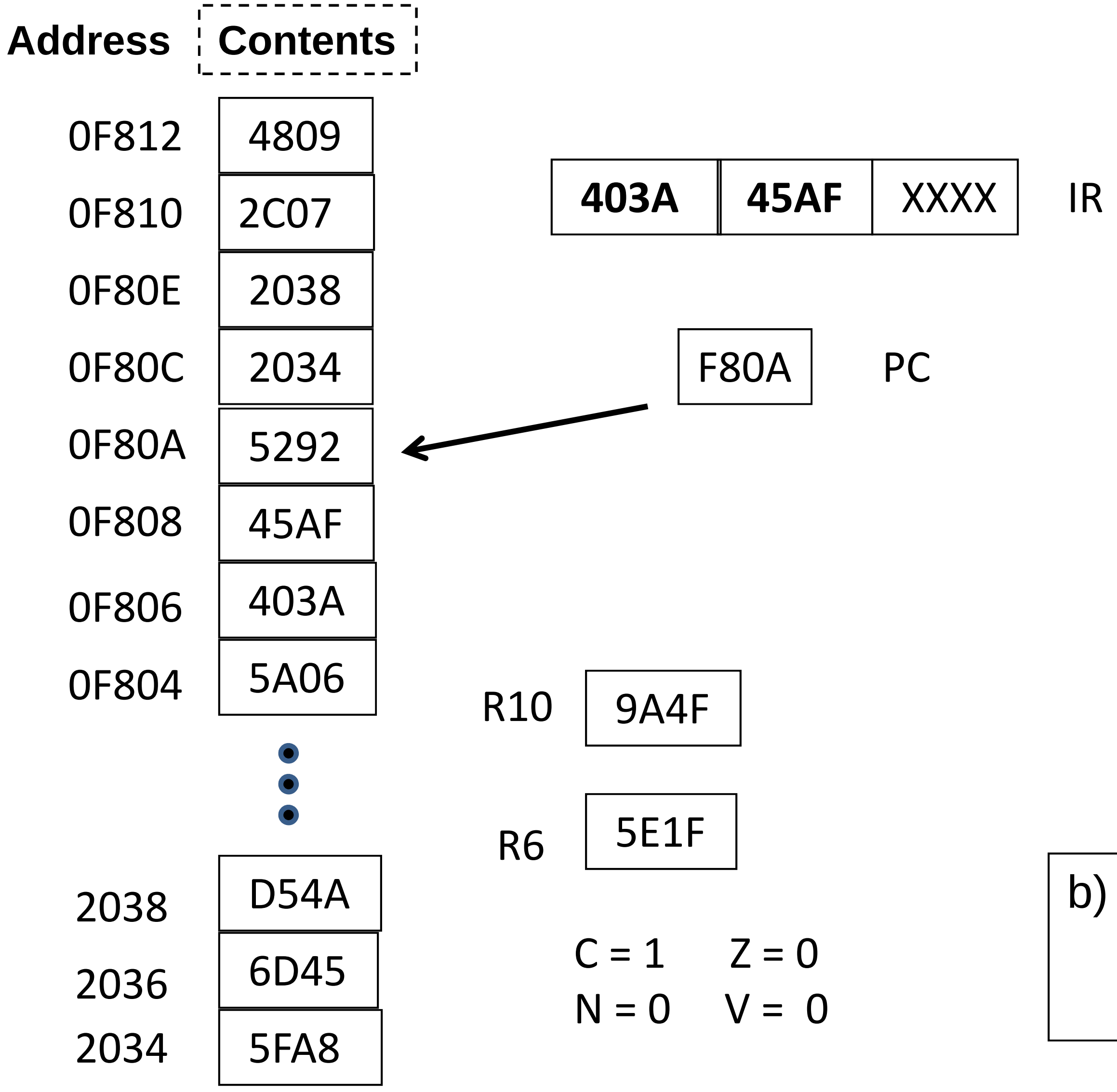


CU determines that the instruction needs the data in memory after the instruction word, so it is necessary to fetch this word <u>(Not an instruction word)</u> to complete decoding.

Therefore, it will fetch the word and place it on the IR before completing decoding.

PC is incremented accordingly

DECODE (2)



Address	Contents
0F812	4809
0F810	2C07
0F80E	2038
0F80C	2034
0F80A	5292
0F808	45AF
0F806	403A
0F804	5A06
⋮	
2038	D54A
2036	6D45
2034	5FA8

403A	45AF	XXXX	IR

PC: F80A

R10: 9A4F

R6: 5E1F

C = 1 Z = 0
N = 0 V = 0

DECODED instruction:

Copy (move) the word 45AF into R10

a) In RTN:

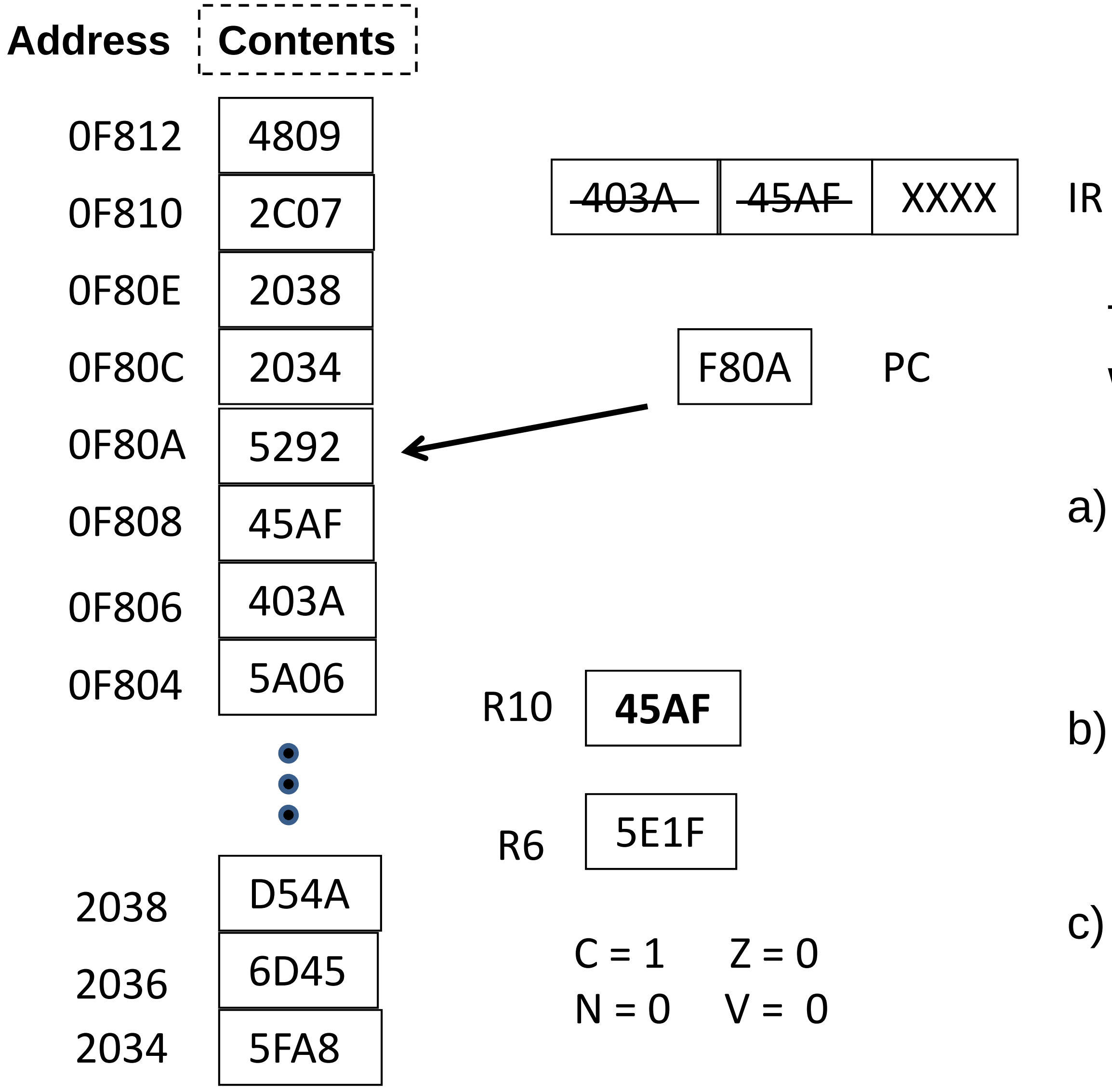
R10 ← 45AF

Also

R10 ← #45AF

b) Decoding is finished and PC is pointing to address following this instruction

Execute



Address	Contents
0F812	4809
0F810	2C07
0F80E	2038
0F80C	2034
0F80A	5292
0F808	45AF
0F806	403A
0F804	5A06
⋮	
2038	D54A
2036	6D45
2034	5FA8

IR: ~~403A~~ ~~45AF~~ XXXX

PC: F80A

R10: **45AF**

R6: 5E1F

C = 1 Z = 0
N = 0 V = 0

The processor executes what decoding indicated:

a) Old contents of destination is lost and has been replaced with new result

b) Flags are not affected by this instruction

c) IR contents is the same, but it is irrelevant

Fetch

Address	**Contents**
0F812	4809
0F810	2C07
0F80E	2038
0F80C	2034
0F80A	5292
0F808	45AF
0F806	403A
0F804	5A06
⋮	⋮
2038	D54A
2036	6D45
2034	5FA8

5292	~~45AF~~	XXXX	IR

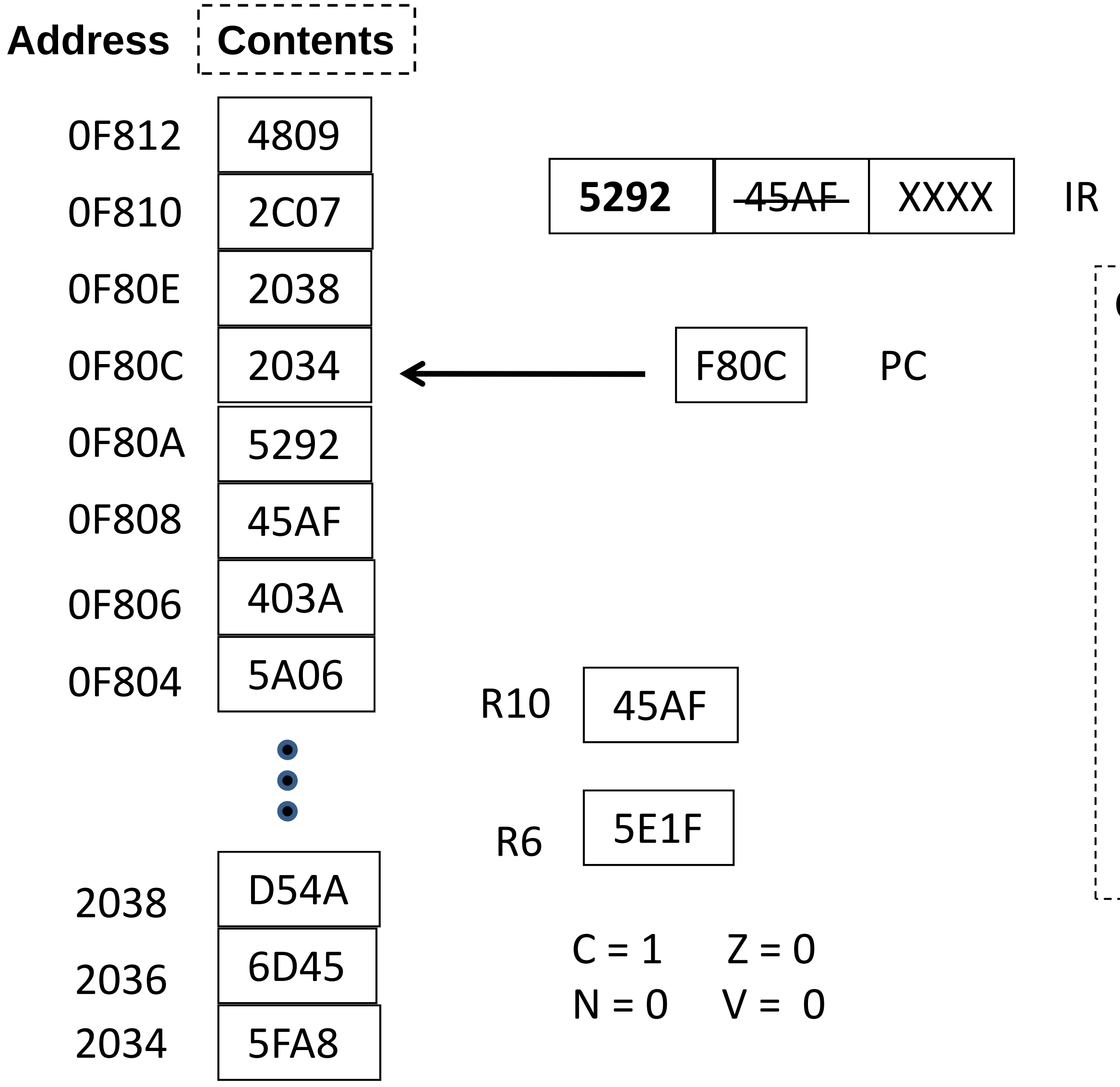
F80C	PC

R10	45AF
R6	5E1F

C = 1   Z = 0
N = 0   V = 0

CU fetches <u>Instruction word</u> and increments PC which is now pointing to next address.

Decode (1)



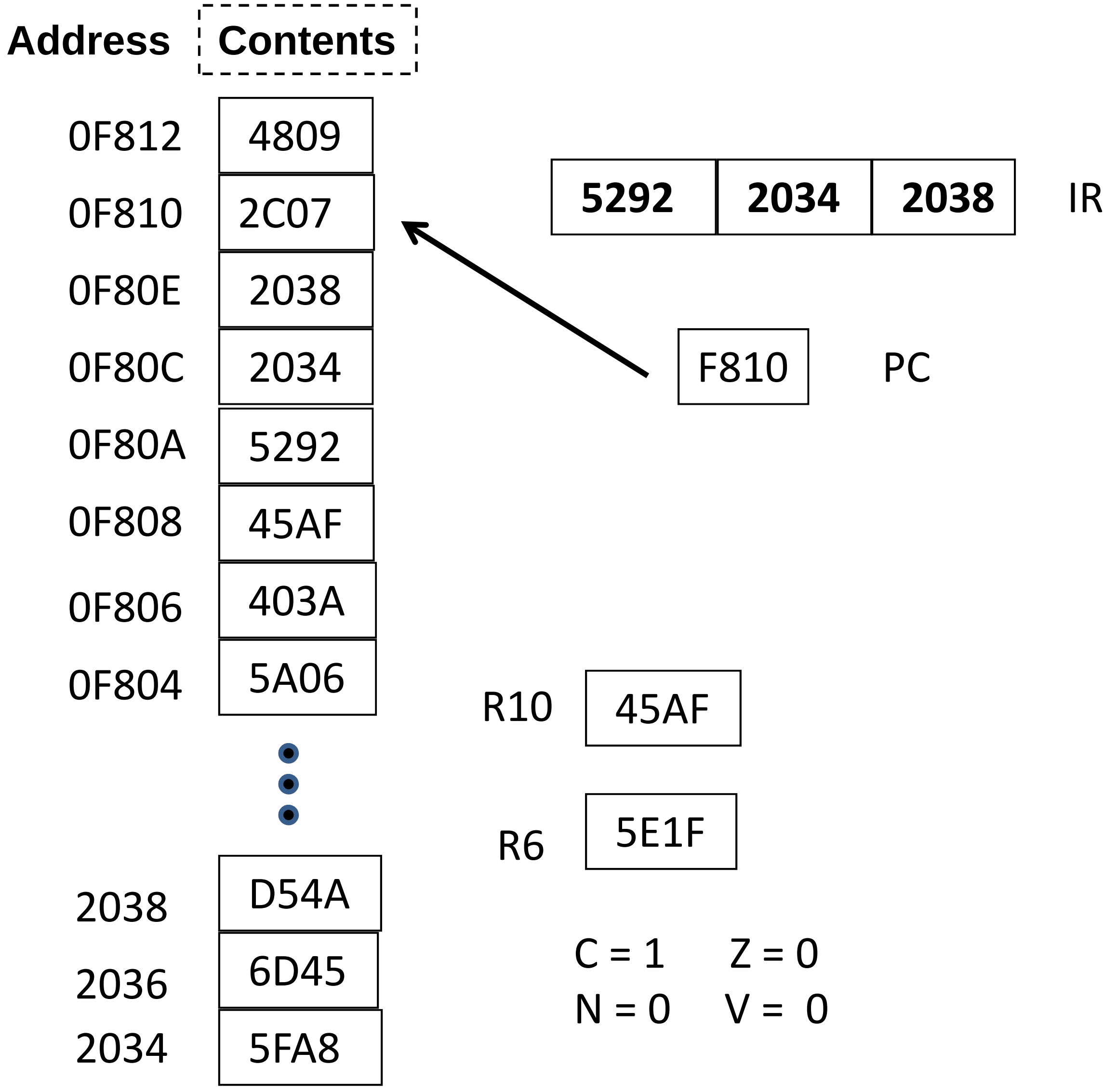
Address	Contents
0F812	4809
0F810	2C07
0F80E	2038
0F80C	2034
0F80A	5292
0F808	45AF
0F806	403A
0F804	5A06
⋮	⋮
2038	D54A
2036	6D45
2034	5FA8

CU determines that the instruction needs data in memory after the instruction word. This time, two words, to complete decoding.

Therefore, it will fetch the words and place them on the IR

PC is incremented accordingly

Decode (2,3)



Address	Contents
0F812	4809
0F810	2C07
0F80E	2038
0F80C	2034
0F80A	5292
0F808	45AF
0F806	403A
0F804	5A06
⋮	
2038	D54A
2036	6D45
2034	5FA8

DECODED instruction:

Add the word in memory with address 2034 to the word in memory with address 2038

a) In RTN:

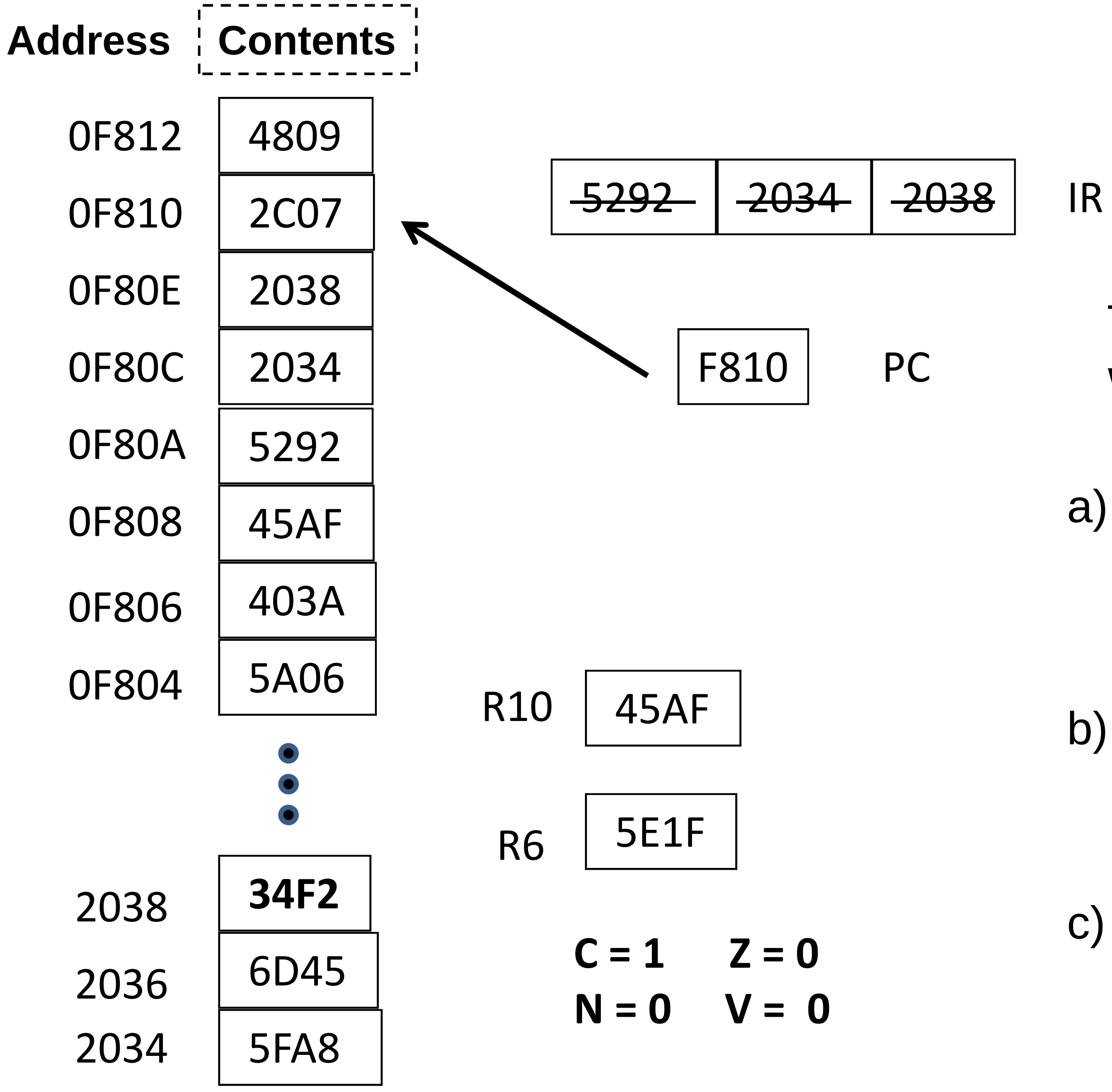
(2038) ← (2034) + (2038)

Also

&2038 ← &2034 + &2038

b) Decoding is finished and PC is pointing to address following this instruction

Execute



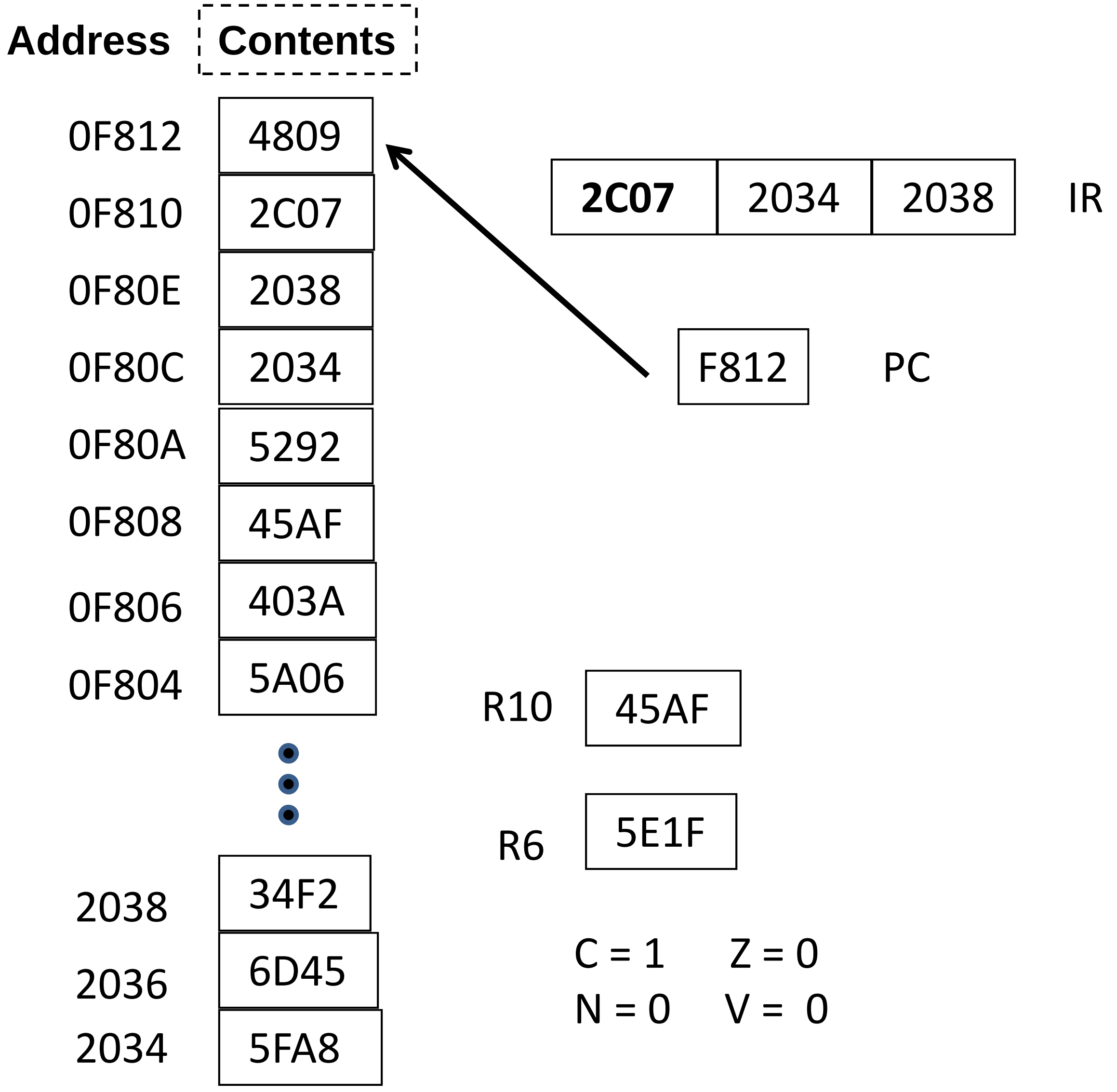
The processor executes what decoding indicated:

a) Old contents of destination is lost and has been replaced with new result

b) Flags are affected by this instruction

c) IR contents is the same, but it is irrelevant

Fetch



Address	Contents
0F812	4809
0F810	2C07
0F80E	2038
0F80C	2034
0F80A	5292
0F808	45AF
0F806	403A
0F804	5A06
⋮	
2038	34F2
2036	6D45
2034	5FA8

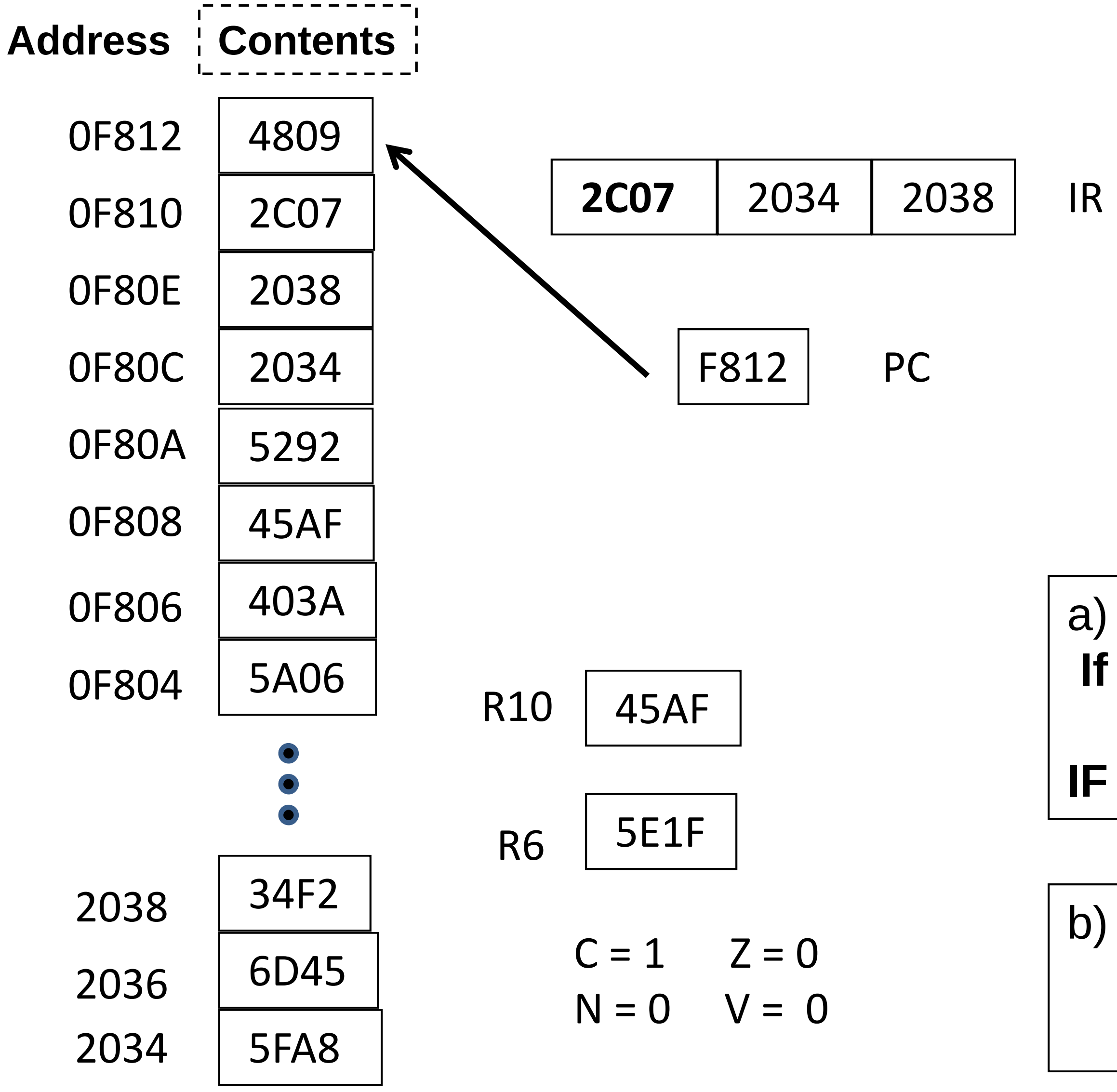
2C07	2034	2038	IR

F812	PC
R10	45AF
R6	5E1F

C = 1 Z = 0
N = 0 V = 0

CU fetches <u>Instruction word</u> and increments PC which is now pointing to next address.

Decode



CU decodes:
IF flag C is set (C=1)
THEN go to instruction at address F820h

Technically, if C is set then

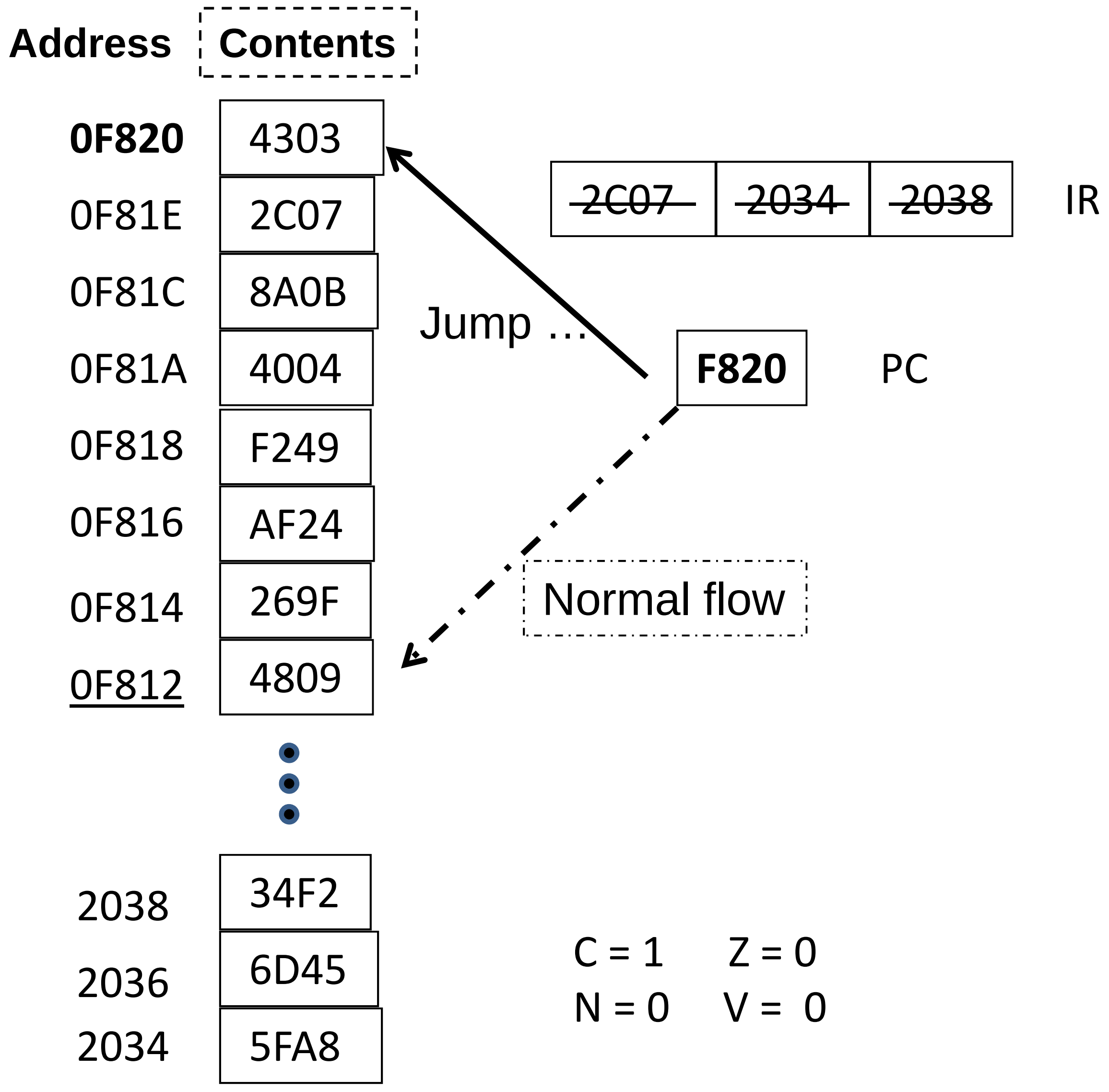
PC ← PC + 2 (0007h)

a) In RTN, express objective;
If C=1, GOTO to F820h or

IF C=1, PC ← F820h

b) Decoding is finished and PC is pointing to address following this instruction

Execute



The execution in this case changes the contents of the PC.

This will cause a JUMP in the sequence of instructions.

The next instruction to be fetched is not the one after the current one.

The instruction does not affect flags.

Important facts to remember

- Instruction can have one or more words
  - **Instruction word:** First word in the set.
  - **Instruction word: Op Code and Addressing modes**
- After the execution state, the PC has the address of the next instruction
- After the decode state, the PC holds the memory address after the current instruction
  - Execution of Program flow instructions may alter PC
  - For other instructions, this is the address of next instruction

Status Register

- Contains flags related to result of execution for some instructions involving ALU and a control Interrupt Flag. All systems include
  - Carry Flag (C)   Zero Flag (Z)
  - Negative Flag (N)   Overflow Flag (V)
  - Interrupt flag (IF) or General Interrupt flag (GIE)
    - Interrupts blocked with IF are called maskable
- Contains group of bits related to system control

Carry flag: Special remarks

- In arithmetic operations, the Carry Flag may have dual function: Carry and Borrow
  - Some MCU's have a separate borrow flag
- Depending on the MCU model (see user guide):
  - C=1 if a borrow is needed in subtraction … or
  - C=0 if a borrow is needed in subtraction
    - **MSP430 adheres to this convention**

Flags and Number comparison (Using A-B)

Comparison	Unsigned Numbers	Signed Numbers
$A = B$	Z=1	Z=1
$A \neq B$	Z=0	Z=0
$A \geq B$	C=1	N=V
$A > B$	C=1 and Z=0	N=V and Z=0
$A < B$	C=0	N$\neq$ V
$A \leq B$	C=0 or Z=1	N$\neq$V or Z=1

Note: This table assumes that C=0 indicates need of borrow in subtraction

Microcomputer Organization

III MSP430 CPU - CPUX

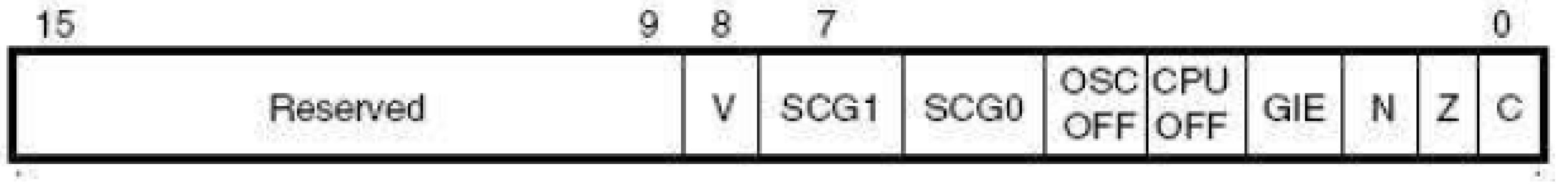
Highlights

- MSP430 offers two architectures:
  - Original MSP430 64K memory, with CPU
  - Extended MSP430X with 1M memory capacity, CPUX
- MSP430X is 100% downward compatible with MSP430
- ALU
  - CPU: 16 bits
  - CPUX: 20 bits

CPU and CPUX registers

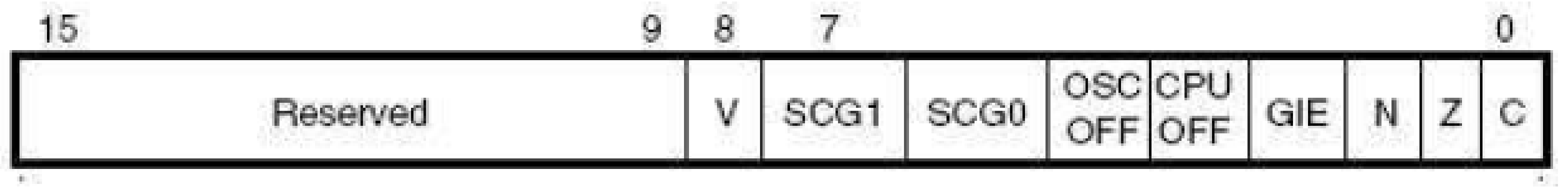
- 16 registers.
  - CPU has 16-bit registers
  - CPUX has 20-bit registers that operate as CPU registers for all CPU instructions .
  - Status Register has 16-bits in both cases.
- Register R0: Program Counter (PC) with bit0=0, hardwired
- Register R1: Stack Pointer (SP) with Bit0 = 0, hardwired
- Register R2: Status Register (SR), 16-bits only
  - Also works as constant generator (CG1)
- Register R3: Constant Generator (CG2)
- Registers R4 to R15: General Purpose generators.

MSP430 Status Register (1/2)



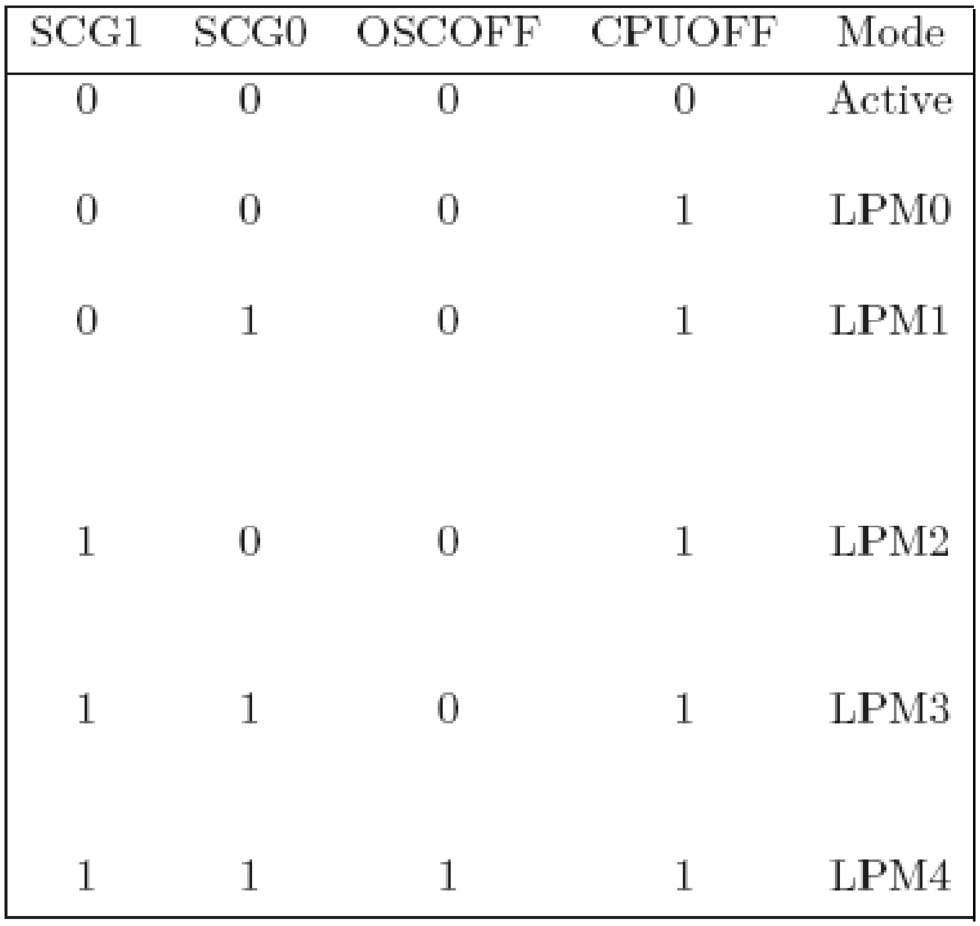
- C: Carry Flag
- Z: Zero Flag
- N: Sign Flag
- GIE: Global Interrupt enable Flag
- V: Overflow Flag

MSP430 Status Register (2/2)



- CPU Off: Turns on and off the CPU
  - CPU Off if CPUOFF=1
- OSCOFF: Turns on and off the Crystal Oscillator
  - Oscillator Off when OSCOFF=1
- SCG1 and SCG0 are combined with CPUOFF and OSCOFF to define the modes of operation

Modes of operation



SCG1	SCG0	OSCOFF	CPUOFF	Mode
0	0	0	0	Active
0	0	0	1	LPM0
0	1	0	1	LPM1
1	0	0	1	LPM2
1	1	0	1	LPM3
1	1	1	1	LPM4

LPM: Low power mode