MEMORY-REFERENCE INSTRUCTIONS

MEMORY-REFERENCE INSTRUCTIONS

- The decoded output Di for i = 0,1, 2, 3, 4, 5, and 6 from the operation decoder that belongs to each instruction.
- The effective address of the instruction is in the address register AR and was placed there during timing signal T2 when I = 0, or during timing signal T3 when I = 1.
- The execution of the memory-reference instructions starts with timing signal T4.

MEMORY-REFERENCE INSTRUCTIONS

- The symbolic description of each instruction is specified in the following table in terms of register transfer notation.

Symbol	Operation	Decoder Symbolic description
AND	D0	AC ← AC ∧ M[AR]
ADD	D1	AC ← AC + M[AR], E ←Cout
LDA	D2	AC ← M[AR]
STA	D3	M[AR] ← AC
BUN	D4	PC ← AR
BSA	D5	M[AR ← PC, PC ← AR + 1
ISZ	D6	M[AR] ← M[AR] + 1, If M[AR] + 1 = 0 then PC ←PC + 1

AND to AC

- This is an instruction that performs the AND logic operation on pairs of bits in AC and the memory word specified by the effective address.
- The result of the operation is transferred to AC.
- The microoperations that execute this instruction are :

$D0\ T4 : DR \leftarrow M[AR]$

$D0\ T5: AC \leftarrow AC \wedge DR, SC \leftarrow 0$

ADD to AC

- The instruction adds the content of the memory word specified by the effective address to the value of AC.
- The sum is transferred into AC and the output carry Cout is transferred to the E (extended accumulator) flip-flop.
- The microoperations needed to execute this instruction are

  D1 T4: DR ← M[AR]

  D1T5: AC ← AC + DR, E ← Cout, SC ← 0

LDA : Load to AC

- This instruction transfers the memory word specified by the effective address to AC.
- The microoperations needed to execute this instruction are

  D2 T4: DR ← M[AR]

  D2 T5: AC ← DR, SC ← 0

STA : Store AC

- This instruction stores the content of AC into the memory word specified by the effective address.
- Since the output of AC is applied to the bus and the data input of memory is connected to the bus, we can execute this instruction with one microoperation:

  D3 T4: $M[AR] \leftarrow AC$, $SC \leftarrow 0$

BUN : Branch Unconditionally

- This instruction transfers the program to the instruction specified by the effective address.
- This instruction is executed with one microoperation:

  D4 T4: PC ← AR, SC ← 0

BSA : Branch and Save Return Address

- This instruction is useful for branching to a portion of the program called a subroutine.
- When executed, the BSA instruction stores the address of the next instruction in sequence (which is available in PC) into a memory location specified by the effective address.
- The effective address plus one is then transferred to PC to serve as the address of the first instruction in the subordinate.
- Micro operation required for this instruction are:

$$M[AR] \leftarrow PC, PC \leftarrow AR + 1$$

BSA : Branch and Save Return Address

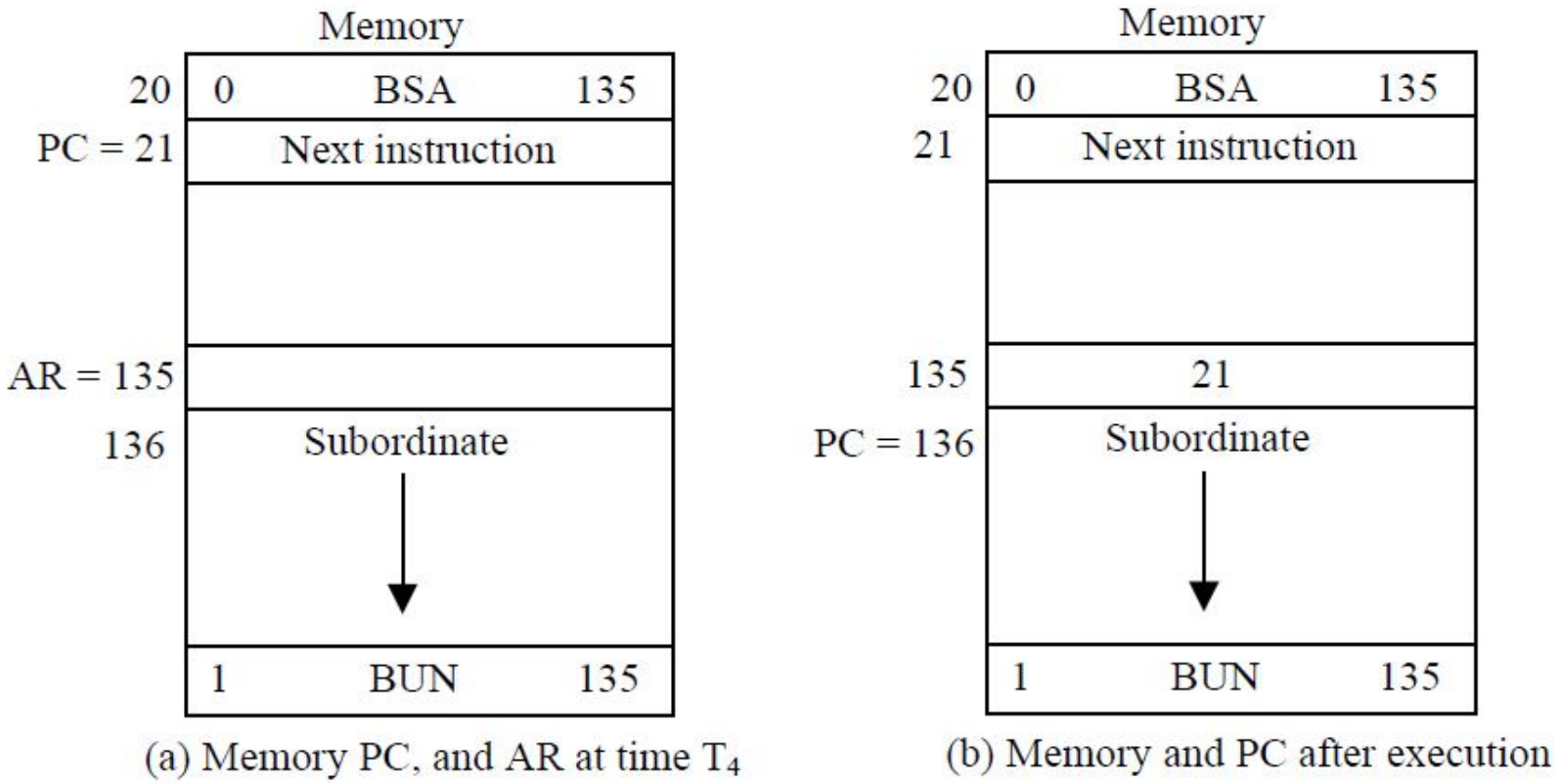

Figure 3.6 Example of BSA instruction execution.

ISZ : Increment and Skip if Zero

- This instruction increments the word specified by the effective address, and if the incremented value is equal to 0, PC is incremented by 1.
- As this negative number is repeatedly incremented by one, it eventually reaches the value of zero. At that time PC is incremented by one in order to skip the next instruction in the program.
- This is done with the following sequence of microoperations:

  D6 T4: DR ← M[AR]

  D6 T5: DR ← DR + 1

  D6 T6 : M[AR] ← DR, if (DR = 0) then (PC ← PC + 1), SC ← 0