Microcomputers I – CE 320

Mohammad Ghamari, Ph.D.
Electrical and Computer Engineering
Kettering University

Announcements

- Lecture on addressing modes is uploaded on blackboard.
- Exercise 2 will be uploaded on blackboard this week.
- Please do not forget to send me an email regarding the peer-teaching section of this course.

Lecture 5: Unconditional Branches

Today’s Goals

Two major goals

- Learn hexadecimal addition and subtraction.
- Use unconditional branches and jump instructions.

Hexadecimal Addition and Subtraction

- For the most part, adding and subtracting in hexadecimal is performed like decimal.

- Only difference is carries or borrows 16 instead of 10.

Hexadecimal Addition

Example 1:

- $5689 + $4574

Hex
0
1
2
3
4
5
6
7
8
9
A
B
C
D
E
F

Hexadecimal Addition

Example 1 (solution):

- $5689 + $4574 = $9BFD

Hexadecimal Addition

Example 2:

- $ADD + $DAD

Hex
0
1
2
3
4
5
6
7
8
9
A
B
C
D
E
F

Hexadecimal Addition

Example 2 (solution):

- $ADD + $DAD = $188A

Hexadecimal Addition

Example 3:

- $2367 + $5FD6

Hex
0
1
2
3
4
5
6
7
8
9
A
B
C
D
E
F

Hexadecimal Addition

Example 3 (solution):

- $2367 + $5FD6 = $833D

Hexadecimal Addition

Example 4:

- $AC22 + 1EE8

Hex
0
1
2
3
4
5
6
7
8
9
A
B
C
D
E
F

Hexadecimal Addition

Example 4 (solution):

- $AC22 + 1EE8 = $CB0A

Hexadecimal Subtraction

Example 1:

- $974B – $587C

Hex
0
1
2
3
4
5
6
7
8
9
A
B
C
D
E
F

Hexadecimal Subtraction

Example 1 (solution):

- $974B – $587C = $3ECF

Hexadecimal Subtraction

Example 2:

- $7788 - $DEF

Hex
0
1
2
3
4
5
6
7
8
9
A
B
C
D
E
F

Hexadecimal Subtraction

Example 2 (solution):

- $7788 - $DEF = $6999

Extending Hexadecimal Numbers

- When a shorter number is added to or subtracted from a longer number, the shorter number must be extended to the same number of digits as the longer number.

- For Unsigned: Always extend by adding 0’s
  - Example: $2357 + $D6 =

- For Signed: Always repeating the sign bits (adding ‘F’s or ‘0’s)
  - Example1 : $2357 + $D6 =
  - Example 2: $2357 +$6D =

Truncating Hexadecimal Numbers

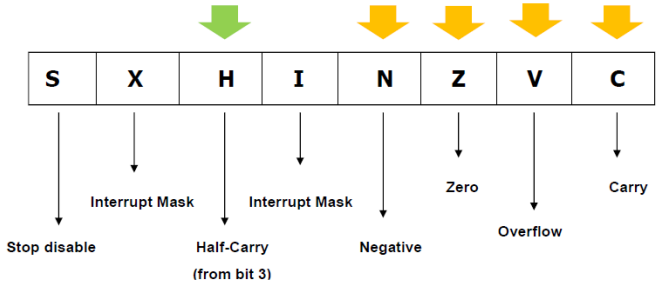
- In microcomputers, it is common practice to represent values using a fewer number of bits to save both time and space.

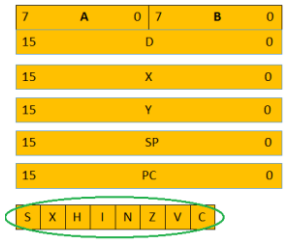
- Shortly, we will see the need to represent the value of a two-byte hexadecimal numbers as a one-byte value if possible.

- For Unsigned: Remove only leading 0s. The remaining number is always valid.
  - Example:
    $00F5
    $10EC ✖

- For Signed: Remove leading 0s and have a positive value or remove leading Fs and have a negative value.
  - Example:
    $0045
    $00F5 ✖
    $FFD1
    $FF66 ✖
    $F280 ✖

Branch Instructions





- Branch instructions **cause program sequence to change** when specific conditions are met.

- Branch instructions can be classified by the type of condition that must be satisfied in order for a branch to be taken.
  - Unary (unconditional) branch*: **Always** branch takes place.
  - Simple branch: Branch **if a condition is satisfied**.
    - A condition is satisfied if certain flags are set.
    - Usually there is a comparison or arithmetic operation to set up the flags before the branch instruction.
  - Unsigned & signed branches: Are taken when a comparison or test of unsigned/signed quantities results in a specific combination of condition code register bits.

Unary Branches		
Mnemonic	**Function**	**Equation or Operation**
BRA	Branch always	1 = 1
BRN	Branch never	1 = 0
Simple Branches		
Mnemonic	**Function**	**Equation or Operation**
BCC	Branch if carry clear	C = 0
BCS	Branch if carry set	C = 1
BEQ	Branch if equal	Z = 1
BMI	Branch if minus	N = 1
BNE	Branch if not equal	Z = 0
BPL	Branch if plus	N = 0
BVC	Branch if overflow clear	V = 0
BVS	Branch if overflow set	V = 1
Unsigned Branches		
Mnemonic	**Function**	**Equation or Operation**
BHI	Branch if higher	C + Z = 0
BHS	Branch if higher or same	C = 0
BLO	Branch if lower	C = 1
BLS	Branch if lower or same	C + Z = 1
Signed Branches		
Mnemonic	**Function**	**Equation or Operation**
BGE	Branch if greater than or equal	N ⊕ V = 0
BGT	Branch if greater than	Z + (N ⊕ V) = 0
BLE	Branch if less than or equal	Z + (N ⊕ V) = 1
BLT	Branch if less than	N ⊕ V = 1

Unconditional branch

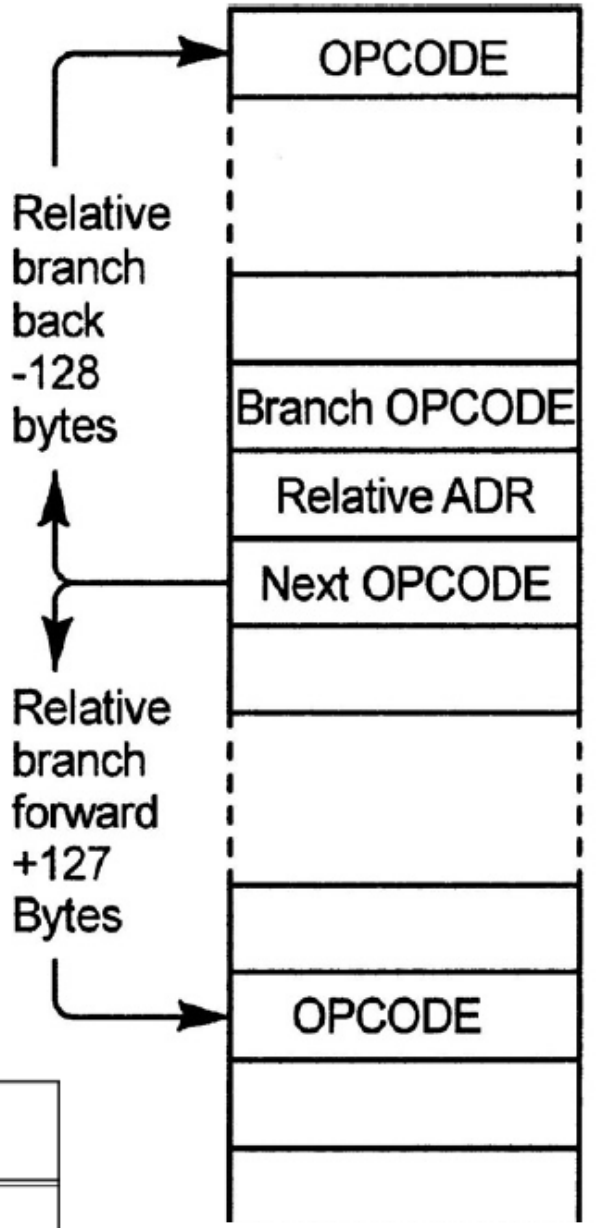
Branch is taken when a specific flag is 0 or 1

Relative Addressing (REL)

Also called PC-relative addressing

Program Counter (PC) is a two-byte register that holds the address of the next instruction to be executed.

- Used **only** by branch instructions that change the program flow (= PC)
  - Short and long conditional branch instructions use the relative mode.
- A **short** branch instructions (ex. BGT,…) consists of an *8-bit* **opcode** and a signed *8-bit* **offset** (distance).
  - The short relative mode can specify a range of $80 (**-128**) ~ $7F (**+127**) from the current PC location.
- A **long** branch instruction (ex. LBEQ,…) consists of an *8-bit* **prebyte**, an *8-bit* **opcode** and a signed *16-bit* **offset** contained in 2 bytes that follow the opcode.
  - The range of the long relative mode is from $8000 (**-32,768**) ~ $7FFF (**+32,767**).

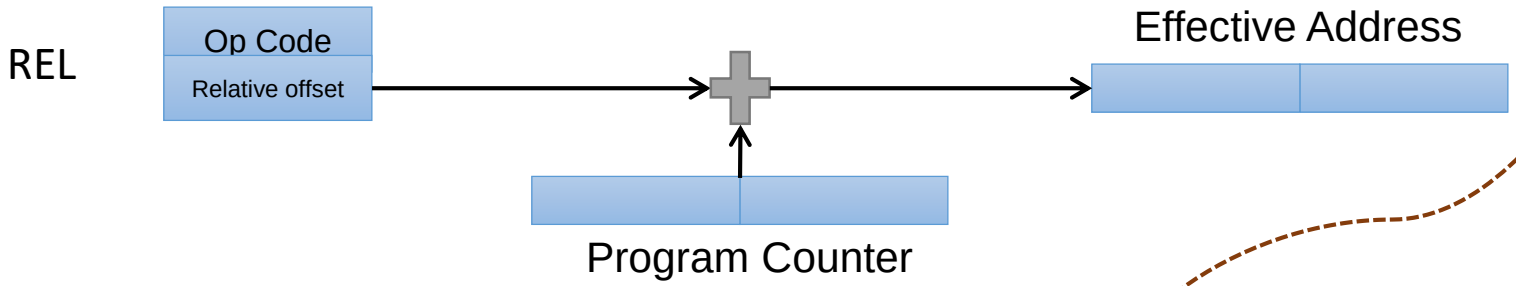


Source Form	Operation	Addr. Mode	Machine Coding (hex)
BGT *rel8*	Branch if Greater Than (if Z + (N ⊕ V) = 0) (signed)	REL	2E rr
LBEQ *rel16*	Long Branch if Equal (if Z = 1)	REL	18 27 qq rr

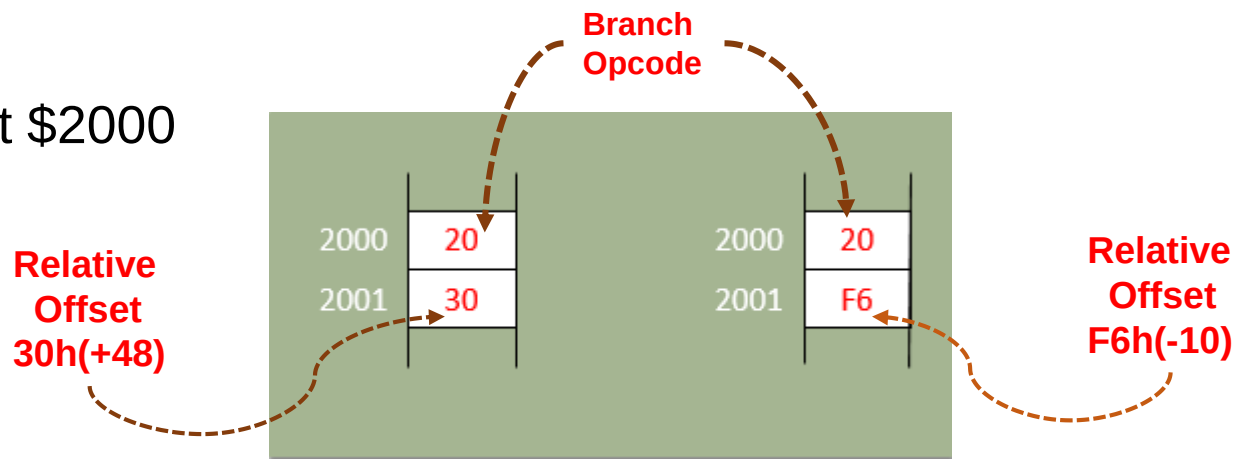
Relative Addressing (REL)

Source Form	Operation	Addr. Mode	Machine Coding (hex)
BRA *rel8*	Branch Always (if 1 = 1)	REL	20 rr

- Effective Address:
  - Add the operand as a signed number to the value in the PC.



- The effective address is loaded into the PC, and the program executes form the new address
- Examples
  - Assuming the instruction is stored at $2000
  - BRA $30
    - Branch always to the instruction 30h forward.
  - BRA -10
    - Branch always to the instruction 10 decimal value backwards

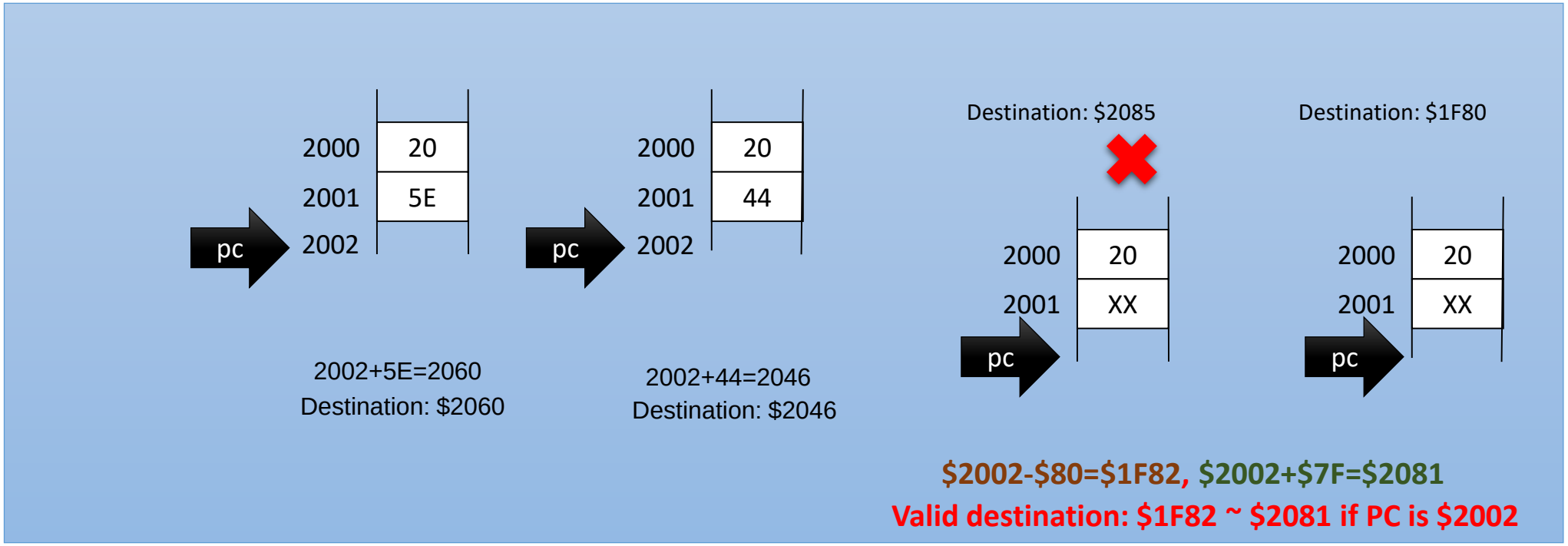


Calculating Branch Destinations

Source Form	Operation	Addr. Mode	Machine Coding (hex)
BRA *rel8*	Branch Always (if 1 = 1)	REL	20 rr

Valid range: -$80 (-128) ~ $7F (127) for short relative mode

- 'Branch' means changing a value of the program counter in the point of view of the microprocessor.
- The destination address can be calculated by adding the operand (either + or -) to the value of the current PC.



Branch and Jump

Instructions

- BRA
  - Branch always
  - Only uses relative addressing (**REL**) with one-byte operands
- LBRA
  - Long Branch always
  - Only uses relative addressing (**REL**) with two-byte operands
- JMP
  - Jump
  - Uses extended(**EXT**) or index addressing (**IDX**)

Questions?

Wrap-up

What we've learned

- Hexadecimal addition and subtraction
- Unconditional branch instructions

What to Come

- Instructions for conditional branches
- HCS12 Assembly language