# Model 4: Translating branch and cmp instructions

Below are two examples of translating a branch instruction to machine code. We don’t show steps that are the same as in Model 3.

Legend: **PC** – program counter – the address of the current instruction

**next PC** – the address of the instruction to go to next

## Example 1.

|  |  |
| --- | --- |
| **Address** | **Source code** |
| 0x00400000 | here: cmp R1, R2 |
| 0x00400004 | beq there |
| 0x00400008 | sub R1, R1, #1 |
| 0x0040000c | b here |
| 0x00400010 | there: or R3, R1, R3 |

## Translating the cmp instruction.

cmp R1, R2

cmp Rn, Rm



Data-processing instruction format

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **cond** | **op** | **I / cmd / S** | **Rn** | **Rd** | **shamt5** | **sh** | **0** | **Rm** |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **cond** | **op** | **I / cmd / S** | **Rn** | **Rd** | **shamt5** | **sh** | **0** | **Rm** |
| 11102 | 002 | 0 1010 1 | 1 | 0 | 0 | 0 | 0 | 2 |

Steps iii and iv as in Model 3.

## Translating the beq instruction.

beq there

beq LABEL

Branch instruction format

|  |  |  |  |
| --- | --- | --- | --- |
| **cond** | **op** | **funct** | **imm24** |

|  |  |  |  |
| --- | --- | --- | --- |
| **cond** | **op** | **funct** | **imm24** |
| 00002 | 102 | 102 | 1 |

|  |  |  |  |
| --- | --- | --- | --- |
| 31:28 | 27:26 | 25:24 | 23:0 |
| **cond** | **op** | **funct** | **imm24** |
| 0000 | 10 | 10 | 0000 0000 0000 0000 0000 0001 |

Step iv as in Model 3.

### Translating the b instruction.

b here

b LABEL

Branch instruction format

|  |  |  |  |
| --- | --- | --- | --- |
| **cond** | **op** | **funct** | **imm24** |

|  |  |  |  |
| --- | --- | --- | --- |
| **cond** | **op** | **funct** | **imm24** |
| 11102 | 102 | 102 | -5 |

|  |  |  |  |
| --- | --- | --- | --- |
| 31:28 | 27:26 | 25:24 | 23:0 |
| **cond** | **op** | **funct** | **imm24** |
| 0000 | 10 | 10 | 1111 1111 1111 1111 1111 1011 |

Step iv as in Model 3.

1. What kind of instruction format does cmp use?

Data-processing format.

1. Looking at step ii of cmp, what fields tell us that this is a cmp instruction? What are the values of those fields?

The i/cmd/s line tells us that it is a cmp instruction. 1010 = cmp. And S = 1

1. Consider the program. For each PC, what is the next PC? If any instructions have two possible next PCs, put a “yes” in that column.

|  |  |  |
| --- | --- | --- |
| PC | next PC | Two possibilities? |
| 0x00400000 | 0x00400004 | no |
| 0x00400004 | 0x00400008 or 0x00400010 | yes |
| 0x00400008 | 0x0040000c | No |
| 0x0040000c | 0x00400000 | No |
| 0x00400010 | Done (end of program) yes but still 0x00400014 |  |

1. For instructions that are ***not*** branch instructions. How are next PC and PC related?

Next PC = PC + 4

1. Here is the RTL of an (unconditional) branch instruction. Of the two ***PC***, which is PC and which is next PC?

PC ← PC + 8 + imm24 << 2

Next PC is left, current PC is right.

1. Rewrite the RTL above into an equation of three variables: PC, next PC, and imm24.

Next PC = PC + 8 + 4\*imm24

4\*imm24 == imm24 << 2

Next, you’ll solve this equation to get the immediate for the two branch instructions in the program.

**Manager:** Split your team into two groups: the first does #29-31. The second does #32-34. Suggested time limit is 5 minutes.

1. What is the PC of the beq instruction?
2. ***When the branch is taken***, what is the next PC of the beq instruction?
3. Use the equation to solve for imm24. Check yourself: does it match the immediate in step ii of the beq translation?
4. What is the PC of the b instruction?
5. What is the next PC of the b instruction?
6. Use the equation solve for imm24. Check yourself: does it match the immediate in step ii of the b translation?

**Check each other.** Come back together as a team and check your answers.

# Read this!

The ***target*** of a branch instruction is the next PC it jumps to if its condition is met. Branch instructions in MIPS use ***relative addressing*** to encode the target address into the immediate field. It is relative because the calculation involves the PC (i.e., the address of the branch instruction).

Now you are going to find the immediate using a different approach.

1. How many *instructions* away from the beq is its target?
2. Compare/contrast the “instructions away” with the immediate for the beq.
3. How many *instructions* away from the b is its target?
4. Compare/contrast the “instructions away” with the immediate for the b.
5. Based on #35-38, describe a method for finding the immediate. Explain how counting instructions in this way relates to the RTL.

PC ← PC + 8 + imm24 << 2

# Exercises

1. In the following program, translate the branch instruction to machine code.

|  |  |
| --- | --- |
| **Address** | **Source code** |
| 0x00100010 | panda: add R1, R1, #1 |
| 0x00100014 | tiger: add R2, R1, R1 |
| 0x00100018 | and R4, R2, #0xFF00 |
| 0x0010001C | and R4, R4, #0xA0 |
| 0x00100020 | or R4, R2, R2 |
| 0x00100024 | cmp R4, R0 |
| 0x00100028 | bne tiger |
| 0x0010002C | j panda |

# Extension questions

1. Determine the maximum distance (in terms of number of instructions) that you can jump forward on a b instruction.
2. Determine the maximum distance (in terms of number of instructions) that you can jump backward on a b instruction.
3. Draw a diagram of instruction memory illustrating your answers to #41 and #42 using an example. You may assume the branch instruction is at address 0x00400010.

|  |  |
| --- | --- |
|  |  |
| ... | ... |
| 0x00400010 | beq ... |
| ... | ... |
|  |  |

1. These questions have demonstrated that an ARM branch instruction cannot jump to far away labels. How might you get around this limitation?