Lecture 3
Conditions and branches

Computing platforms

Novosibirsk State University
University of Hertfordshire

D. Irtegov, A.Shafarenko

2018

CdM-8 flag semantics

- N – sign bit of the result. Used for signed comparison
- C – carry bit of the result. Used for unsigned comparison
- Z – result is zero. Used for signed, unsigned and bitwise comparison
- V – signed overflow (sign loss). Can be used to catch errors
- V is also needed for *correct* signed comparison

C and unsigned subtraction/comparison again

- Subtraction ⇔ adding 2’complement
- When the result < 0, C is 0
- 1-255 = 1+0000 0001 = 2
- When the result > 0, C is 1
- 3-2 = 11+1111 1110 = 1+C

Full list of CdM-8 branch instructions

condition	test	interpretation
eq/z	Z	equal, equal to zero / Zero is set
ne/nz	¬Z	not equal, not zero, Zero is clear
hs/cs	C	unsigned higher or same / Carry is set
lo/cc	¬C	unsigned lower / Carry is clear
mi	N	negative (minus)
pl	¬N	positive or zero (plus)
vs	V	oVerflow is set
vc	¬V	oVerflow is clear
hi	C∧¬Z	unsigned higher
ls	¬C∨Z	unsigned lower or same
ge	(N∧V) ∨ (¬N∧¬V)	greater than or equal, greater than or equal to zero
lt	(N∧¬V) ∨ (¬N∧V)	less than, less than zero
gt	(¬Z∧N∧V) ∨ (¬Z∧¬N∧¬V)	greater than, greater than zero
le	(Z∨N∧¬V) ∨ (¬N∧V)	less than or equal, less than or equal to zero

Figure 5.4: Control conditions.

More about branches

- In typical assembler, branch is like goto statement.
- You must invent label names and jump to labels
- Typical equivalent of
  `if (condition) { then-block } else {else-block)`
  requires one comparison, two labels, one branch and one jump
- (unconditional branch)

Condition calc

b[!cond] $1

    Then-block

Br $2

$1: Else-block

$2: …

CdM-8 assembler has richer syntax

If

    Calc condition

is cond

    Then-block

Else

    Else-block

Fi

Real example

```
if
        tst r0
is z
        ldi r1, 10
        add r1, r0
else
        shla r0
fi
```

- Consult tome.pdf for syntax for complex conditions
- (it is not so elegant)

Loops

```
# r2=r0*r1 (assuming r1 is non-negative)
        clr r2
while
        tst r1
stays gt
        add r0, r2
        dec r1
wend
```

Post-condition loop

```
# find a zero
        ldi r0, array-1
# Initialise r0 to point to the cell before the first element of the array.
do
        inc r0          # point r0 to the next element
        ld r0, r1       # read the element into r1
        tst r1          # examine it
until z                 # if r1 is 0 then exit, otherwise continue
```

Nesting of if’s and loops is possible

- You can use them like blocks in high-level languages
- You do not need to invent label names
- You do not need to worry about correct nesting
- Much harder to write spaghetti code (than with raw branches)
- This is why CdM-8 assembly is called Platform 3 ½
- Actually, it is much simplier to implement than you probably think
- It is all described in tome.pdf
- Beware: in some exercises using structural statements is explicitly prohibited