EECE 218
Microcontrollers

Branches, math & logic instructions
Branch instructions

● Two variants, both use REL addressing
  » ‘Short branch’ : B___
    – Range: -128..+127 of the next instruction
  » ‘Long branch’ : LB___
    – Range -32768..+32767 of the next instruction (anywhere)

● More variants:
  » Loop branches:
    – Dec/Inc/Test counter and branch if eq/neq (‘short’)
  » Bit condition branches
    – Branch if selected bit set/clear (‘short’)

Branch instructions – Simple

- Unconditional: BRA(lways), BRN(ever)

- Single bit test in CCR:
  - N(egative) BMI/BPL Result -
  - Z(ero) BEQ/BNE Result 0
  - o(V)erflow BVS/BVC Wrong sign
  - C(arr)ey BCS/BCC Carry over

- Long variants:
  - LBRA,LBRN,LBMI,….LBCC

- Signed/unsigned conditional branches:
  - USE CPU REF GUIDE!!!
Branch instructions – Simple

- Before branch: some compare between a R(egister) and M(emory)
  - E.g. CPX #$1234 → R=X, M value = $1234

- Signed conditional branches (signed R, M)
  - R > M BGT/BLE
  - R >= M BGE/BLT
  - R <= M BLE/BGT
  - R < M BLT/BGE

- Unsigned conditional branches
  - R > M BHI/BLS
  - R >= M BHS/BLO
  - R <= M BLS/BHI
  - R < M BLO/BHS

'greater/lesser'

'higher/lower'
Conditional branch example:

if (A > 32) L1: - Unsigned
  CMPA  #32
  BHI   L1

if (B < -2) L2: - Signed
  CMPB  #-2
  BLT   L2

if (MAX <= A) L3 - Unsigned
  CMPA  MAX
  BHS   L3
Loop branches

- **Common form:**
  - \{d,i,t\}b{eq,ne} cntr,rel
  - cntr = counter in A,B,D,X,Y, or SP
  - rel = (short) relative label
  - d = decrement (-1), i = increment (+1), t = test (0?)

- **Meaning:**
  - Decrement/Increment/Test counter and if it is equal/not equal to ZERO, take the branch

- **Example: Execute loop 100 times**
  - LDAB #100
  - L <loop body>
  - DBNE B,L
Bit condition branches

- **Common form:**
  
  br{clr,set} opr,mask,rel
  
  - Checks if interesting bits are clear (0) or set (1)
  - opr = operand in memory (DIR, EXT, INDX)
  - mask = byte value, has 0-s for uninteresting bits, 1-s for interesting bits
  - rel = relative branch target ('short')

- **Meaning:**
  
  - Check the selected ‘interesting’ bits of the operand, and take the branch if all the selected bits are set (brset) or if they are clear (brclr)
Bit condition branches

Example:

```assembly
loop    brclr $66,$01,loop
```
Reads [$0066] and tests the least significant bit. If that bit is zero, it loops.

```assembly
wait    brset $77,$F0,wait
```
Reads [$0077] and tests the most significant 4 bits in that byte. If all of them are set, it loops.
Problem: write a program that multiplies/divides by powers of 2.

1. OPER(and): ← modified
2. ARG(ument): ← specifies power of 2
   If ARG > 0 : + power of 2 (*2,*4,*8,…)
   If ARG < 0 : - power of 2 (/2,/4,/8,…)
   If ARG = 0 : 1 --- do nothing.
Powers of 2

- Example: 25 multiplied by 4
  - %0001 1001 = 25
  - Shift left: %0011 0010 = 48+2 = 50
  - Shift left: %0110 0100 = 96 + 4 = 100
  - Each shift left means multiplication by 2!
  - Shift right: %0000 1100 = 12
  - Shift right: %0000 0110 = 6
  - Each shift right means division by 2!

In binary, shifting means multiplication/division by powers of 2.
Shift/rotate instructions

- Arithmetic('signed')/Logic ('unsigned')/Rotate
- Arithmetic shift:
  - ASL (A,B,mem,D) (*2)
  - ASR (A,B,mem) (/2, sign bit is preserved!)
Shift/rotate instructions

- Logic shift:
  - LSL (A,B,mem,D) (*2)
  - LSR (A,B,mem,D) (/2)
Shift/rotate instructions

- Rotate left: (spec math ops)
  » ROL (A,B,mem)
  » ROR (A,B,mem)
Powers of 2 problem

ARG =? 0
   Y → Done
   N → ARG >? 0
       Y → Negate ARG
           N → LSL OPER
               N → DEC/BNE ARG
                   Y → Done
                   N → LSR OPER
                       N → DEC/BNE ARG
                           Y → Done
Powers of 2 - Code

PO2  LDAA ARG ; LDA sets Z bit
       BEQ DONE
       BMI DIV ; LDA sets N bit
L1  LSL OPER
       DBNE A,L1 ; Decr & br if != 0
       BRA DONE
DIV  NEGA ; Negate A (2’s compl.)
L2  LSR OPER
       DBNE A,L2
DONE RTS ; Subroutine return
ARG  DS.B 1 ; Mem for ARG
OPER DS.B 1 ; Mem for OPER