.global maxS16 @ (int16\_t x, int16\_t y) // returns the maximum of x, y

.global maxU32 @ (uint32\_t x, uint32\_t y) // returns the maximum of x, y

.global isGreaterThanU16 @ (uint16\_t x, uint16\_t y) // returns 1 if x>y, 0 else

.global isGreaterThanS16 @ (int16\_t x, int16\_t y) // returns 1 if x>y, 0 else

.global shiftRightS32 @ (int32\_t x, uint8\_t p) // returns x >> p = x\*2^(-p) for 0..31

.global shiftU16 @ (uint16\_t x, int8\_t p) // return x\*2^p for p = -31..31

.global isMultOf4U32 @ bool isMultOf4U32(uint32\_t x) // returns 1 if x is an integer multiple of 4

.global isEqualU16 @ bool isEqualU16(uint16\_t x, uint16\_t y) // returns 1 if x=y, 0 if x!=y

1)

int16\_t maxS16(int16\_t x, int16\_t y) // returns the maximum of x, y

maxS16:

CMP R0, R1

MOVLT R0,R1

BX LR

2)

uint32\_t maxU32(uint32\_t x, uint32\_t y) // returns the maximum of x, y

maxU32:

CMP R0, R1

MOVLO R0, R1

3)

bool isGreaterThanU16(uint16\_t x, uint16\_t y) // returns 1 if x>y, 0 else

isGreaterThanU16:

CMP R0, R1

MOVLS R0, #1

MOVHI R0, #0

BX LR

4)

bool isGreaterThanS16(int16\_t x, int16\_t y) // returns 1 if x>y, 0 else

isGreaterThanS16:

CMP R0, R1

MOVLE R0, #0

MOVGT R0, #1

BX LR

5)

uint16\_t shiftU16(uint16\_t x, int8\_t p) // return x\*2^p for p = -31..31

shiftU16:

CMP R1, #0

BMI right\_shift

MOV R0, R0, LSL R1

BX LR

right\_shift:

MVN R1,R1

ADD R1,R1, #1

MOV R0, R0, LSR R1

BX LR

6) bool isBitSetU32(uint32\_t x, uint32\_t bit) // returns 1 if the requested bit is set in x, 0 else

isBitSetU32:

MOV R2, #1

SUB R1, R1, #1

MOV R1, R2, LSL R1

ANDS R0, R0, R1

MOVEQ R0, #0

MOVNE R0, #1

BX LR

7)

bool isMultOf4U32(uint32\_t x) // returns 1 if x is an integer multiple of 4, 0else (e.g, 0, 4, 8, 12, 16, ...

are integer multiples of 4

isMultOf4U32:

ANDS R0, R0, #3

MOVEQ R0, #1

MOVNE R0, #0

BX LR

8) bool isEqualU16(uint16\_t x, uint16\_t y) // returns 1 if x=y, 0 if x!=y

isEqualU16:

CMP R0, R1

MOV R0, #0

MOVEQ R0, #1

BX LR