CSE2312-002, 004 (Fall 2021)
Homework #3

Notes:
With this homework, we start writing assembly functions for the RPi 3b/3b+.

All numbers are in base-10 unless otherwise noted.

If part of a problem is not solvable, explain why in the answer area.

The target date to complete this homework set is October 5, 2021.

This homework set will not be graded, but please solve all of the problems to prepare for the quizzes and exams.

1. For each of these C functions, specify the ARM7 register(s) in which each argument is passed and result is returned.

a. uint32_t fn4(uint16_t a, uint32_t b, int8_t c, uint32_t d)

a is passed in: R0

b is passed in: R1

c is passed in: R2

d is passed in: R3

the result is returned in: R0

b. uint64_t fn2(uint64_t a, uint64_t b)

a is passed in: R0:R1

b is passed in: R2:R3

the result is returned in: R0:R1

2. Write 1-5 line assembly functions that implement the following C functions:

a. uint64_t addU32_U64(uint32_t x, uint32_t y) // returns x+y

b. uint64_t addU64(uint64_t x, uint64_t y) // returns x+y

c. int32_t convertS8ToS32(int8_t x) // converts 8-bit signed value to 32-bits signedfs
   (don't overthink this... it is a one-line function)

d. int32_t convertU16ToS32(uint16_t x) // converts 16-bit unsigned value to 32-bits signed

e. int16_t maxS16(int16_t x, int16_t y) // returns the maximum of x, y

f. uint32_t maxU32(uint32_t x, uint32_t y) // returns the maximum of x, y

g. bool isGreaterThanU16(uint16_t x, uint16_t y) // returns 1 if x>y, 0 else

h. bool isGreaterThanS16(int16_t x, int16_t y) // returns 1 if x>y, 0 else

i. int32_t shiftRightS32 (int32_t x, uint8_t p) // returns x >> p = x*2^(-p) for p = 0..31

j. uint16_t shiftU16(uint16_t x, int8_t p) // return x*2^p for p = -31..31

k. bool isEqualU16(uint16_t x, uint16_t y) // returns 1 if x=y, 0 if x!=y

```
.global

addU32_U64          @ (uint32_t x, uint32_t y) // returns x+y
addU64              @ (uint64_t x, uint64_t y) // returns x+y
convertS8ToS32      @ (int8_t x) // converts 8-bit signed value to 32-bits
convertU16ToS32     @ (uint16_t x) // converts 16-bit unsigned value to 32-bits signed
maxS16              @ (int16_t x, int16_t y) // returns the maximum of x, y
maxU32              @ (uint32_t x, uint32_t y) // returns the maximum of x, y
isGreaterThanU16    @ (uint16_t x, uint16_t y) // returns 1 if x>y, 0 else
isGreaterThanS16    @ (int16_t x, int16_t y) // returns 1 if x>y, 0 else
shiftRightS32       @ (int32_t x, uint8_t p) // returns x >> p = x*2^(-p) for 0..31
shiftU16            @ (uint16_t x, int8_t p) // return x*2^p for p = -31..31
isEqualU16          @ (uint16_t x, uint16_t y) // returns 1 if x=y, 0 if x!=y


.text

@ a. uint64_t addU32_U64(uint32_t x, uint32_t y) // returns x+y
addU32_U64:
      MOV         R2, R1
      MOV         R1, #0
      ADDS        R0, R0, R2
      ADC         R1, R1, #0
      BX  LR

@ b. uint64_t addU64(uint64_t x, uint64_t y) // returns x+y
addU64:
      ADDS        R0, R0, R2
      ADC         R1, R1, R2
      BX  LR

@ c. int32_t convertS8ToS32(int8_t x) // converts 8-bit signed value to 32-bits
convertS8ToS32:
      BX  LR

@ d. int32_t convertU16ToS32(uint16_t x) // converts 16-bit unsigned value to 32-bits signed
convertU16ToS32:
      BX  LR

@ e. int16_t maxS16(int16_t x, int16_t y) // returns the maximum of x, y
maxS16:
      CMP         R0, R1
      MOVLT       R0,R1
      BX  LR

@ f. uint32_t maxU32(uint32_t x, uint32_t y) // returns the maximum of x, y
maxU32:
      CMP         R0, R1
      MOVLO       R0, R1
```

```
@ g. bool isGreaterThanU16(uint16_t x, uint16_t y) // returns 1 if x>y, 0 else
isGreaterThanU16:
      CMP         R0, R1
      MOVLS       R0, #0
      MOVHI       R0, #1
      BX  LR

@ h. 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

@ i. int32_t shiftRightS32 (int32_t x, uint8_t p) // returns x >> p = x*2^(-p) for p = 0..31
shiftRightS32:
      ASR         R0, R0, R1
      BX  LR

@ j. uint16_t shiftU16(uint16_t x, int8_t p) // return x*2^p for p = -31..31
shiftU16:
      CMP         R1, #0
      BLT right_shift
      LSL         R0, R0, R1
      BX  LR

  right_shift:
      MOV         R2, #0
      SUB         R1, R2, R1
      LSR         R0, R0, R1
      BX  LR

@ k. bool isEqualU16(uint16_t x, uint16_t y) // returns 1 if x=y, 0 if x!=y
isEqualU16:
      CMP         R0, R1
      MOVEQ       R0, #1
      MOVNE       R0, #0
      BX  LR
```