Exercise 1

(a)0x00000020, cacheable, RAM

(b)0x00000020, noncacheable, RAM

(c)0x1F800001, noncacheable, SFRs

(d)0x1FC00111, cacheable, boot flash

(e)0x1D001000, cacheable, flash

Exercise 3

(a) PORTB: 0-15

PORTC: 12-15

PORTD: 0-11

PORTE: 0-7

PORTF: 0, 1, 3, 4, 5

PORTG: 2, 3, 6, 7, 8, 9

RE0: Pin 60

(b)

unimplemented: 5, 6, 7, 11, 13, 14, 15, 17-31

implemented: bit 0: INT0EP

bit 1: INT1EP

bit 2: INT2EP

bit 3: INT3EP

bit 4: INT4EP

bit 8, 9, 10: TPC <2:0>

bit 12: MVEC

bit 16: SS0

Exercise 7

A .o file contains compiled object code which is machine code together with the names of the functions and other objects that the file contains. Object files are processed by the linker to produce the final executable.

A .hex file are used for hexadecimal source files which contain settings and configuration information.

Many .o files archiver to .a files which is object files libraries and link it with other files to .elf files. With applying the binary-to-hexadecimal (xc32-bin2hex) which converts binary files (from the 32-bit linker) to Intel hex format files and will reduce the storage.

Exercise 8

(a)

```
##############################################################################
# Call main. We do this via a thunk in the text section so that
# a normal jump and link can be used, enabling the startup code
# to work properly whether main is written in MIPS16 or MIPS32
# code. I.e., the linker will correctly adjust the JAL to JALX if
# necessary
##############################################################################
and     a0,a0,0
and     a1,a1,0
la      t0,_main_entry
jr      t0
nop

.end _startup



   # Call main
##############################################################################
la          t0,main
jalr        t0
nop
```

(b)

```
ffffffffbf800000 A RTCCON1
ffffffffbf800004 A RTCCON1CLR
ffffffffbf800008 A RTCCON1SET
ffffffffbf80000c A RTCCON1INV
ffffffffbf800010 A RTCCON2
```

(c)

```
#define _SPI2STAT_SPIRBF_POSITION          0x00000000
#define _SPI2STAT_SPIRBF_MASK              0x00000001
#define _SPI2STAT_SPIRBF_LENGTH            0x00000001

#define _SPI2STAT_SPITBF_POSITION          0x00000001
#define _SPI2STAT_SPITBF_MASK              0x00000002
#define _SPI2STAT_SPITBF_LENGTH            0x00000001

#define _SPI2STAT_SPITBE_POSITION          0x00000003
#define _SPI2STAT_SPITBE_MASK              0x00000008
#define _SPI2STAT_SPITBE_LENGTH            0x00000001

#define _SPI2STAT_SPIRBE_POSITION          0x00000005
#define _SPI2STAT_SPIRBE_MASK              0x00000020
#define _SPI2STAT_SPIRBE_LENGTH            0x00000001

#define _SPI2STAT_SPIROV_POSITION          0x00000006
#define _SPI2STAT_SPIROV_MASK              0x00000040
#define _SPI2STAT_SPIROV_LENGTH            0x00000001

#define _SPI2STAT_SRMT_POSITION            0x00000007
#define _SPI2STAT_SRMT_MASK                0x00000080
#define _SPI2STAT_SRMT_LENGTH              0x00000001

#define _SPI2STAT_SPITUR_POSITION          0x00000008
#define _SPI2STAT_SPITUR_MASK              0x00000100
#define _SPI2STAT_SPITUR_LENGTH            0x00000001

#define _SPI2STAT_SPIBUSY_POSITION         0x0000000B
#define _SPI2STAT_SPIBUSY_MASK             0x00000800
#define _SPI2STAT_SPIBUSY_LENGTH           0x00000001

#define _SPI2STAT_TXBUFELM_POSITION        0x00000010
#define _SPI2STAT_TXBUFELM_MASK            0x001F0000
#define _SPI2STAT_TXBUFELM_LENGTH          0x00000005

#define _SPI2STAT_RXBUFELM_POSITION        0x00000018
#define _SPI2STAT_RXBUFELM_MASK            0x1F000000
#define _SPI2STAT_RXBUFELM_LENGTH          0x00000005
```

18 bits are defined

SPIRBF: 1bit

SPITBF: 1bit

SPITBE: 1bit

SPIRBE: 1bit

SPIROV: 1 bit

SRMT: 1 bit

SPITUR: 1bit

SPIBUSY: 1bit

TXBUFELM<4:0>: 5bits

RXBUFELM<4:0>: 5bits

Yes, it shows in data sheet.

Exercise 9

TRISDSET = 0b1100

TRISDCLR = 0b100010

TRISDINV = 0b10001