Engineering FIRST YEAR

Part 1A Paper 3: Electrical and Information Engineering

DIGITAL CIRCUITS AND INFORMATION PROCESSING

SOLUTIONS TO EXAMPLES PAPER 4

1. (a) The program counter holds the address of the next instruction to be executed and is incremented after each instruction is implemented. The working register holds the data the PIC is working on at the current time (a bit like the memory on a simple calculator).

   The STATUS register stores the results of the previous calculation (the carry, zero and half-carry flags) as well as power down, time out and register bank information.

   The GPIO register holds data either input or output to pins on the chip. The TRISIO register holds the information as to (and is used to set) which particular bits in the GPIO register are inputs and which are outputs.

   (b) Tri-state logic has an enable input which allows the logic output to have a third state other than high or low - the output can be left floating (effectively unconnected) so that it can be defined by other circuits connected there.

   All circuits connected to the data bus are equipped with tri-state outputs. At any one time only one circuit drives the bus, and other circuits monitor the resulting voltages. The microprocessor can drive the bus during a *write* cycle, and a memory chip can drive the bus during a *read* cycle. The bus can thus be said to be bi-directional.

   (c) Erasable Programmable Read Only Memory is used to store start-up programs, and data that does not change and needs to be preserved when the power is turned off.

   (d) The Fetch-Decode-Execute cycle refers to the cycle by which an instruction is implemented by the microprocessor. The processor first **fetches** the instruction from the address stored in the PC. The fetched instruction is then **decoded** so that it can be interpreted by the microprocessor. Once decoded, the instruction is **executed** and the PC incremented so that it contains the address of the next instruction.

2. (a)

```
movlw 0x30 ;     moves the (hex) number 30 into W
movwf 0x30 ;     moves contents of W (0x30) into locn 0x30
movlw 30 ;       moves the decimal number 30 into W
addwf 0x30 ;     adds contents of W to content of 0x30
```

   So contents of 0x30 will be the sum of 0x30 and 30 which is 78.

(b)

```
          movlw 4 ;           moves 4 (decimal)
          movwf 0x20 ;        into 0x20 via W
          clrf 0x30 ;         sets contents of 0x30 to 0
    lab   incf 0x30 ;         increments the contents of 0x30
          decfsz 0x20 ;       decrements 0x20, skips next instruction if zero
          goto lab ;          jumps to start of loop (at label lab)
          sleep ;             enters sleep mode
```

0x30 is incremented each time the loop occurs – which is 4 times. So at the end of the programme the contents of 0x30 is 4.

3. (a) The highest address the memory will respond to is 0xC0FF, the lowest is 0xC000. This is a range of 256 bytes. The size of the memory is therefore 2048 bits.

(b) The required circuit is shown in Figure 1

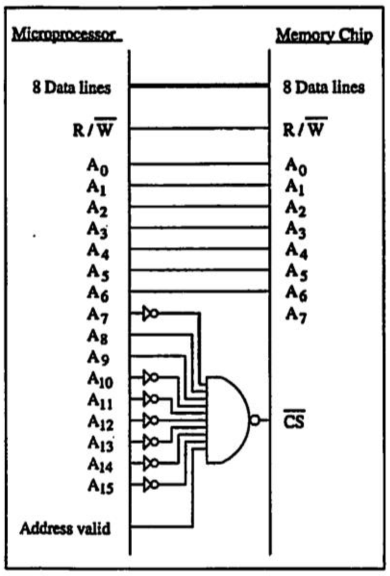
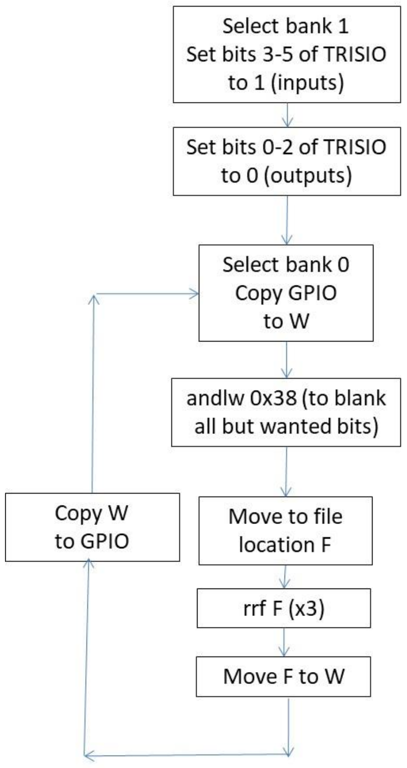
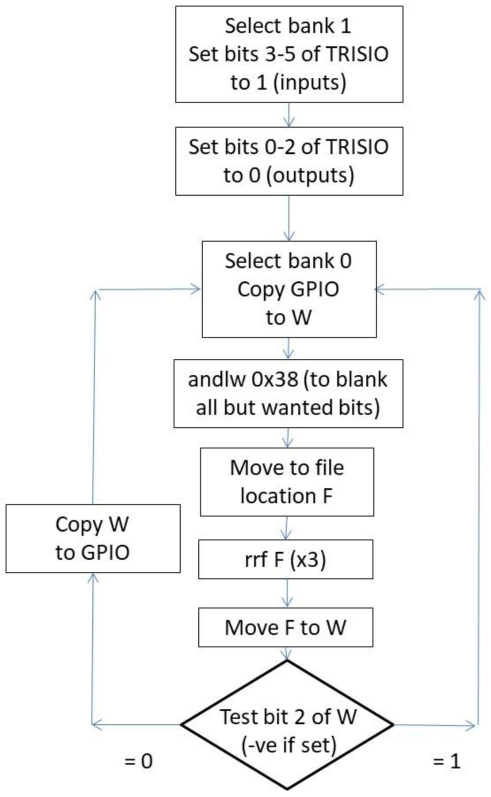

Microprocessor
Memory Chip
8 Data lines
8 Data lines
$R/\overline{W}$
$R/\overline{W}$
A_0 A_1 A_2 A_3 A_4 A_5 A_6 A_7 A_8 A_9 A_{10} A_{11} A_{12} A_{13} A_{14} A_{15}
A_0 A_1 A_2 A_3 A_4 A_5 A_6 A_7
$\overline{CS}$
Address valid

Figure 1

4 (a) The flow diagrams are shown in Figure 2(a) for part (a) and Figure 2(b) for part (b). Please note that formally bank 1 needs to be set to manipulate TRISIO and then bank 0 to use GPIO (students have been told this in lectures but it is not stressed).


Select bank 1
Set bits 3-5 of TRISIO to 1 (inputs)
Set bits 0-2 of TRISIO to 0 (outputs)
Select bank 0
Copy GPIO to W
andlw 0x38 (to blank all but wanted bits)
Move to file location F
rrf F (x3)
Move F to W
Copy W to GPIO


Select bank 1
Set bits 3-5 of TRISIO to 1 (inputs)
Set bits 0-2 of TRISIO to 0 (outputs)
Select bank 0
Copy GPIO to W
andlw 0x38 (to blank all but wanted bits)
Move to file location F
rrf F (x3)
Move F to W
Test bit 2 of W (-ve if set)
= 0
= 1
Copy W to GPIO

(a) (b)

Figure 2

5 Bit 0 (the C flag) of the STATUS register is set when there is a carry out of the 8 bit word. Bit 1 (DC flag) is set when there is a carry from the lower to the higher hex digit (nibble) in the addition, and bit 2 (Z flag) is set when the answer is zero.

Hex			Unsigned					
A	B	Sum	A	B	Sum	C	DC	Z
$7F	$01	$80	127	1	128	0	1	0
$C0	$0A	$CA	192	10	202	0	0	0
$27	$D9	$00	39	217	0	1	1	1
$08	$0F	$17	8	15	23	0	1	0
$80	$80	$00	128	128	0	1	0	1

6
```
      movlw 0x38 ;       sets bits 0-2 of TRISIO to 0 (outputs)
      movwf TRISIO ;      and bits 3-5 to 1 (inputs)
lab   movf GPIO, W ;     load GPIO into W (NB bits 3-5)
      movwf 0x20;        moves data in 0x20
      rrf 0x20;
      rrf 0x20;          puts bits 3,4 into 1,2 of 0x20 (NB has multiplied by 2 as 2 rrfs, not 3)
      movf 0x20, W;      and back into W
      movwf GPIO ;       original bits 3,4 in 1,2 of GPIO (ie multiplies by 2)
      bcf GPIO, 0 ;      clears bit 0 of GPIO
      goto lab ;
```

NB a slightly less clunky approach would be to use the rrf W,W command on the data directly, rather than moving into and out of a dummy memory location. However, students have not been taught this.

7 Assembled code found in data book.

Instruction	Prog Memory Locn	Assembled code	Notes
movlw 4	0x0005	11 00XX 0000 0100	literal value last 8 bits
movwf 0x20	0x0006	00 0000 1010 0000	Data mem location 0x20 or 100000
clrf 0x30	0x0007	00 0001 1011 0000	Data mem location 0x30 or 110000
lab incf 0x30	0x0008	00 1010 1011 0000	
decfsz 0x20	0x0009	00 1011 1010 0000	Result in F so d=1
goto lab	0x000A	10 1000 0000 1000	lab instruction in 0x08 or 1000
sleep	0x000B	00 0000 0110 0011	

8 (a)

```
        movlw 22 ;          puts 22 into W (1~)
        movf 0x80, W ;      copies contents of 0x080 into W (1~)
        subwf 0x81, W ;     subtracts W from contents of 0x081 (1~)
        btfss 0x03, 2 ;     tests bit 2 (Z flag) of STATUS registers, skips next if set
        call sr ;           call subroutine sr (ie when not equal) (2~)
        sleep ;             enter standby mode (1~)
        ……
sr      movwf 0x82 ;        copies contents of W into 0x82 (1~)
        return              (2~)
```

Numbers are equal:
1 clock period for first three instructions, for btfss, the conditional test is true and so this takes 2 clock periods. Sleep instruction, 1 period. Tota1 6 clock periods (0.6 µs).

Numbers are not equal:
1 clock period for first three instructions, for btfss, the conditional test is not true and so this takes 1 clock period. Subroutine call and return both take 2 clock periods while the instruction in the subroutine takes 1. Sleep instruction, 1 period. Tota1 10 clock periods (1.0 µs).

(b)

```
        movlw 20 ;            puts 20 into (1~)
        movwf 0x30 ;          0x30 (1~)
lab     decf 0x30 ;           decrements contents of 0x30 (1~)
        btfss STATUS, 2 ;     tests Z flag and skips goto if set (1~  if not set, 2~ if set )
        goto lab ;            jump to lab (2~)
        sleep ;               processor into sleep mode (1~)
```

There are two clock cycles before the loop. The loop consists of 4 cycles and this is carried out 19 times. On the 20th, the goto instruction is skipped so three cycles (on for the dec instruction and 2 for the btfss instruction where conditional test is true for this case) , followed by one for the sleep instruction. Total cycles are 2 + (19x4) + 3 + 1 = 82 cycles.
Note, there are several other ways of approaching this question – e.g. using the decfsz instruction, or even 81 nop commands followed by the sleep command (though the latter goes against the spirit of the question).

R V Penty
Easter 2019