Homework #8

Connor Maller

1) Given the C program program for serial communication. Answer the questions for parts a and b below.

```
#include <avr/io.h>

unsigned char cleared = 1<<UDRE;

int main()
{
        int i;
        DDRD = 0b00000010;
        PORTD = 0xFF;
        UCSRB = (1<<RXEN) | (1<<TXEN);
        UCSRC = (1<<UCSZ1) | (1<<UCSZ0) | (1<<URSEL);
        UBRRL = 0x33;
        while(1)
        {
                while(!(UCSRA & cleared));
                UDR = 'G';
        }

        return 0;
}
```

a. Is serial transmission or reception or both modes enabled? Which serial operation(s) is(are) performed in the program (transmission/reception/both)? Circle the program instruction(s) that you used to determine your answer.
b. Is polling or an interrupt used in this program for serial communication? Circle the program instruction(s) that you used to determine your answer.

Polling

2) Below is a C program that outputs square waves using Timers 0 and 2. Create a Microchip Studio workspace (GCC C Executable Project for the ATmega32) and copy the C code into the C workspace. Answer the questions for parts a-e on the following page.

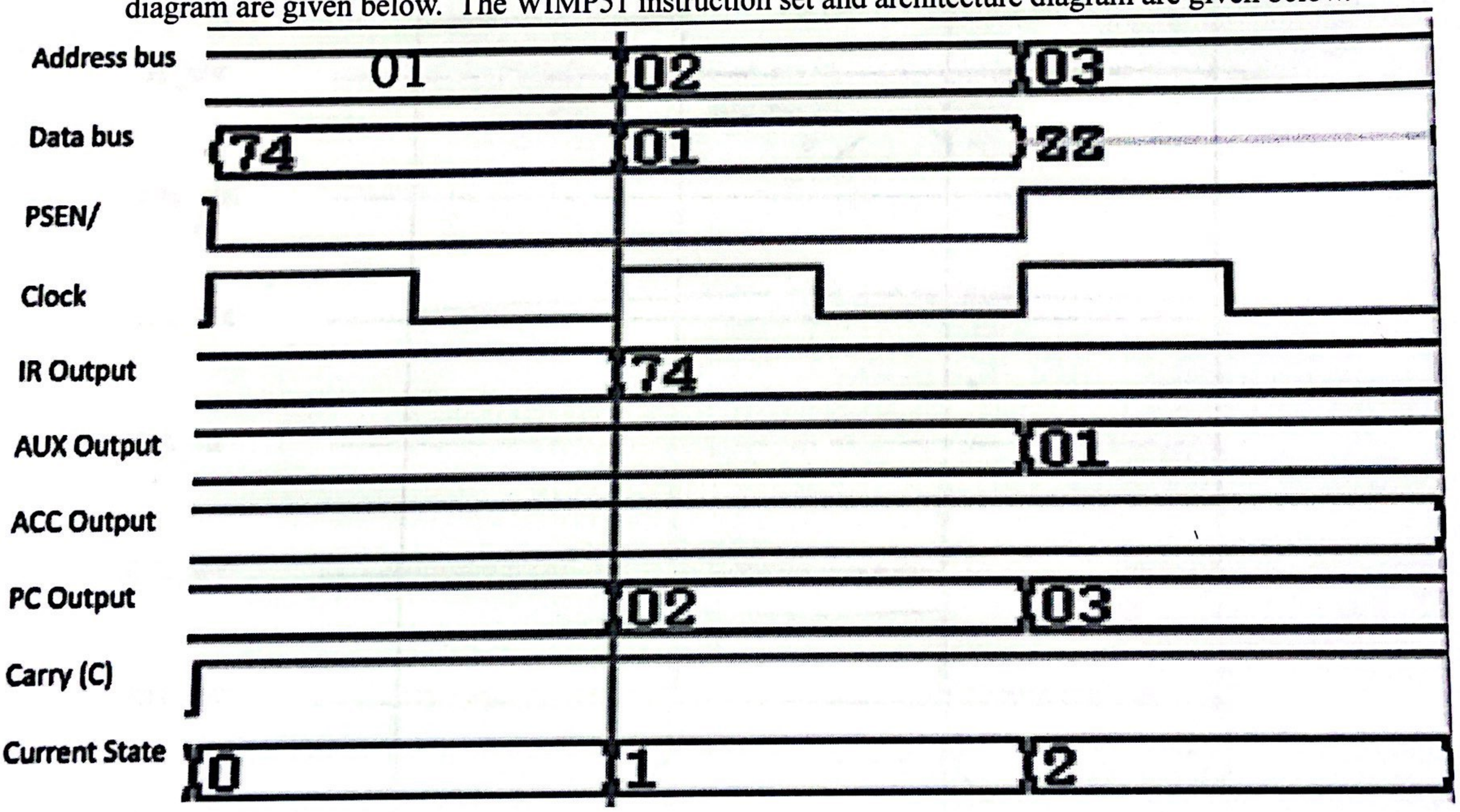
```
#include <AVR/IO.H>
#include <AVR/INTERRUPT.H>

ISR(TIMER0_OVF_vect);
ISR(TIMER2_OVF_vect);


int main()
{
        DDRB |= 0B01100000;
        TIMSK = (1<<TOIE0) | (1<<TOIE2);
        TCNT0 = -30;
        TCNT2 = -60;
        TCCR0 = 0X05;
        TCCR2 = 0x02;
```

```
        sei();
        while(1);
        return 1;
}

ISR(TIMER0_OVF_vect)
{
        TCNT0 = -30;
        PORTB ^= 0X40;
}

ISR(TIMER2_OVF_vect)
{
        TCNT2 = -60;
        PORTB ^= 0X20;
}
```

a) From the Disassembly window of the workspace, identify the interrupt code for the interrupt routines for Timers 0 and 2. Submit a screenshot of showing the C/assembly code for both interrupt service routines.

b) Describe differences between the C and Assembly code implementations for the waveforms generated using Timers 0 and 2 (excluding version of the program is in C and the translated version is in assembly). Is the assembly code an efficient translation of the C code? Explain.
the addition of the registers being popped & pushed & jumped. These operation one not in C. It is an efficent translation of the C.

3) Given the following timing signals for the WIMP51 for an instruction cycle. For the Current State signal, note that 0 = fetch state, 1 = decode state, 2 = execute state. The instruction set and architecture diagram are given below.

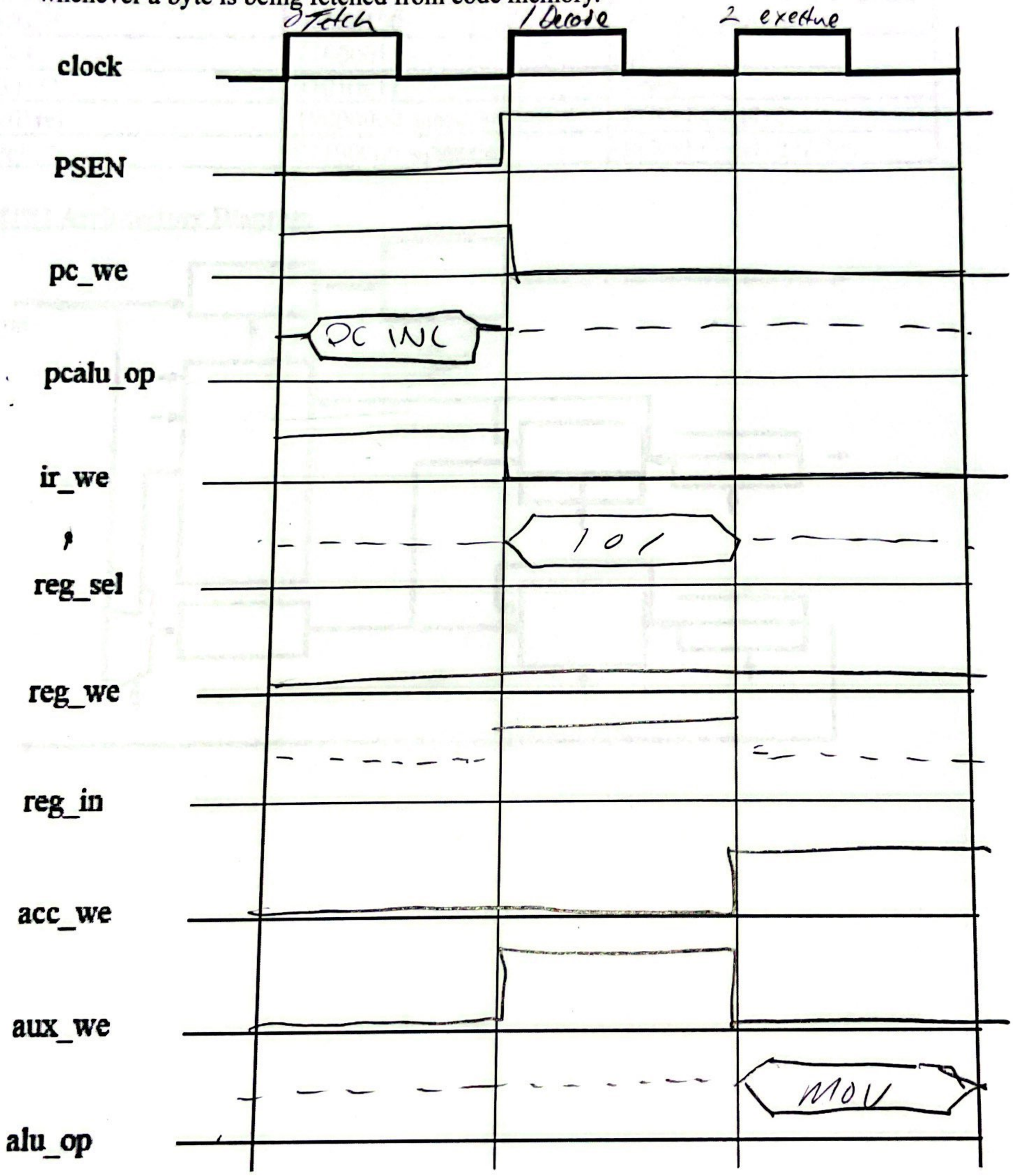


MOV A, #0x01 0111 0100 = 0x74 = MOV A, #D

Determine the instruction being executed during the instruction cycle.

4) Draw the timing diagram for the following control signals on the diagram below when the WIMP51 is executing the instruction ___MOV A, R5_____. Label the fetch, decode and execute parts of the instruction cycle on the clock portion of the signal below.

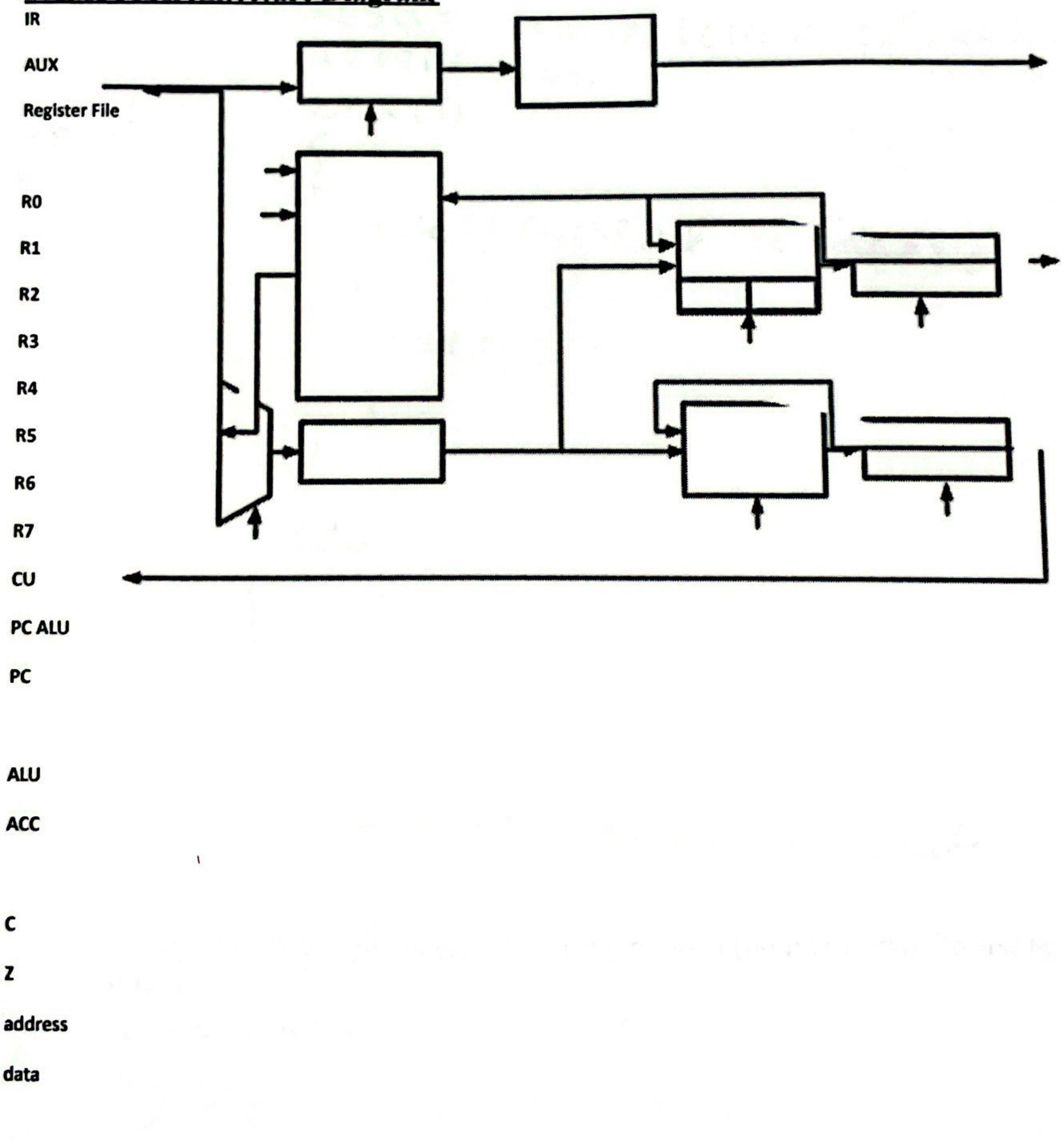
The dotted lines have been drawn as a graphical reference only and do not reflect the value of the corresponding signal. If you think a particular control signal is of no significance during a particular time period, indicate this opinion by using a **dashed** line to denote the signal as a "don't care" for that time period. Give "command" values for pcalu_op and alu_op; for example, indicate the value of "pcalu_op" as "PC_INC" or "PC_ADD" rather than representing this command by a numeric code. If you are not sure about a particular control signal, make an educated guess, indicate that the value you have marked is a guess, and explain why you made this particular guess. Remember, PSEN/ pulses low whenever a byte is being fetched from code memory.



Summary of WIMP51 Inst Set

ASM code	Machine code	Meaning
MOV A, #D	01110100 dddddddd	A<=D
ADDC A,#D	00110100 dddddddd	C,A<=A+D+C
MOV Rn, A	11111nnn	Rn<=A
MOV A, Rn	11101nnn	A<=Rn
ADDC A, Rn	00111nnn	C,A<=A+Rn+C
ORL A, Rn	01001nnn	A<=A OR Rn
ANL A, Rn	01011nnn	A<= A AND Rn
XRL A, Rn	01101nnn	A<=A XOR Rn
SWAP A	11000100	A<=$A_{(3-0)}$&$A_{(7-4)}$
CLR C	11000011	C<=0
SETB C	11010011	C<=1
SJMP rel	10000000 aaaaaaaa	PC<=PC+rel+2
JZ rel	01100000 aaaaaaaa	PC<=PC+rel+2 if Z=1

WIMP51 Architecture Diagram



pcalu_op

acc_we

pc_we

aux_we

reg_in

ir_we

reg_we

reg_sel

reg_sel

0

1

acc_out

PSEN

34a) $x = \left(\frac{10 \cdot 10^6}{16(9600)}\right) - 1 = 64.11 \approx 64 = 0x400 = UBRR$

c) $x = \left(\frac{10 \cdot 10^6}{16(1200)}\right) - 1 = 519.83 \approx 520 = 0x208 = UBRR$

36)
```
#include <avr/io.h>
int main()
{

UCSRB=(1<<TXEN);
UCSRC=(1<<UCSZ1)|(1<<UCSZ0)|(1<<URSEL);
UBRRL=0x40;
while(1)
{

  while(!(UCSRA&(1<<URXE)))
  {

   UDR='2';
  }
}
Return 0;
}
```

→ more on next sheet

5) Chapter 11: ~~7, 15, 30, 34 a,c,~~ 36-polling method (no interrupts), 36-use serial interrupts (XTAL = 10 MHz)
Problems are below.

7) False, neither are compatible

15) 4,000,000 bits

30) USART Baud Rate Register

```
#include <avr/io.h>
#include <avr/interrupt.h>

ISR(USART0_UDRE_vect);
int main()
{
   UCSRB = (1<<TXEN)|(1<<UDRIE);
   UCSRC = (1<<UCSZ1)|(1<<UCSZ0)|(1<<URSEL);
   UBRRL = 0x40;
   sei();
   while(1);
   return 0;
}

ISR(USART0_UDRE_vect)
{
  UDR = 'z';
}
```