CSSE2010/CSSE7201
Learning Lab 11

Introduction to AVR 324 Dev Board

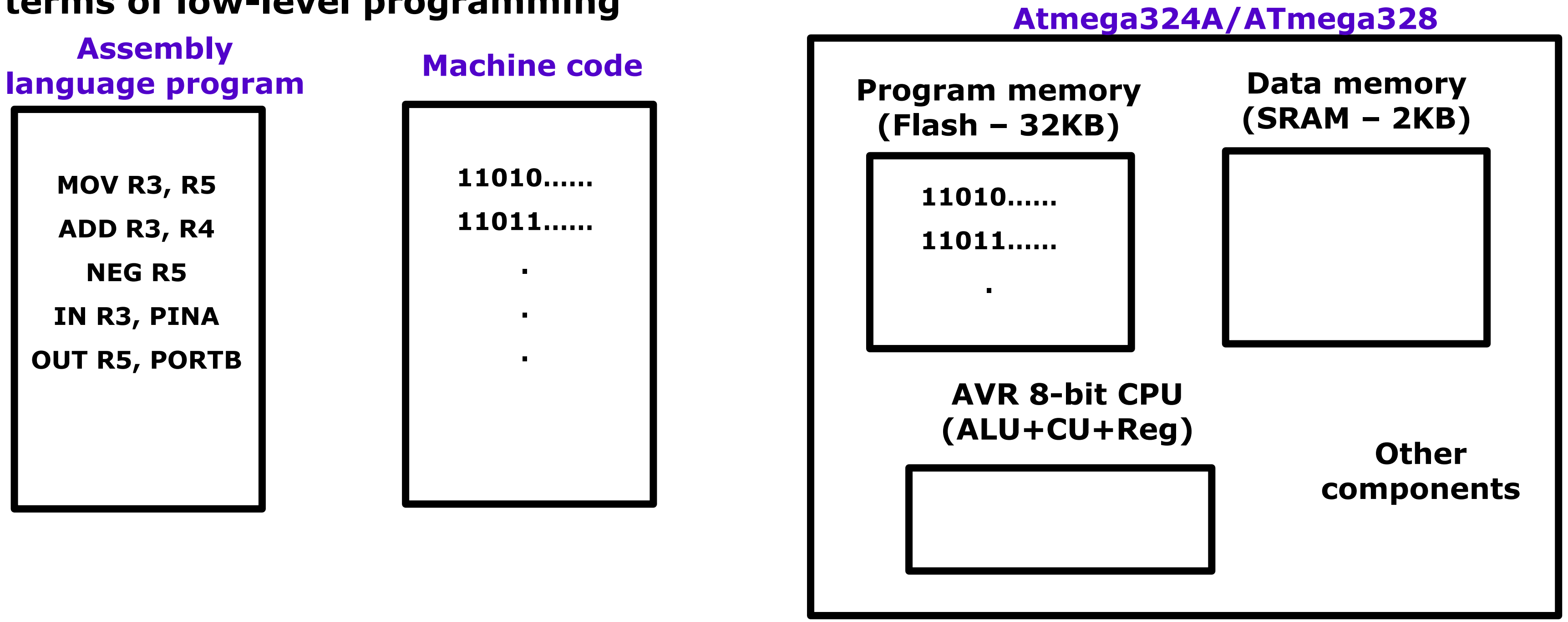
School of Information Technology and Electrical Engineering
The University of Queensland

Today

- AVR I/O ports
  - Register Numbers
  - Port, pin and data direction registers
  - IN and OUT instructions
- Using and programming the 324 Dev board

Recap from lectures

A very simplified description on what is happening inside the microcontroller in terms of low-level programming



AVR I/O Registers & General Purpose I/O Ports

- From lectures
  - ATmega324A has 224 I/O registers
    - Accessible at memory addresses 32 (0x20) to 255 (0xFF)
      - Use ld (load) and st (store) instructions
- First 64 I/O registers (0 to 63) can also be accessed using "in" and "out" instructions
- See pages 636 to 639 of datasheet (or I/O register summary) for I/O register details
  - Many registers reserved (not used)
  - Note general purpose I/O port registers
    - DDRA, PORTA, PINA, DDRB, PORTB, PINB, …

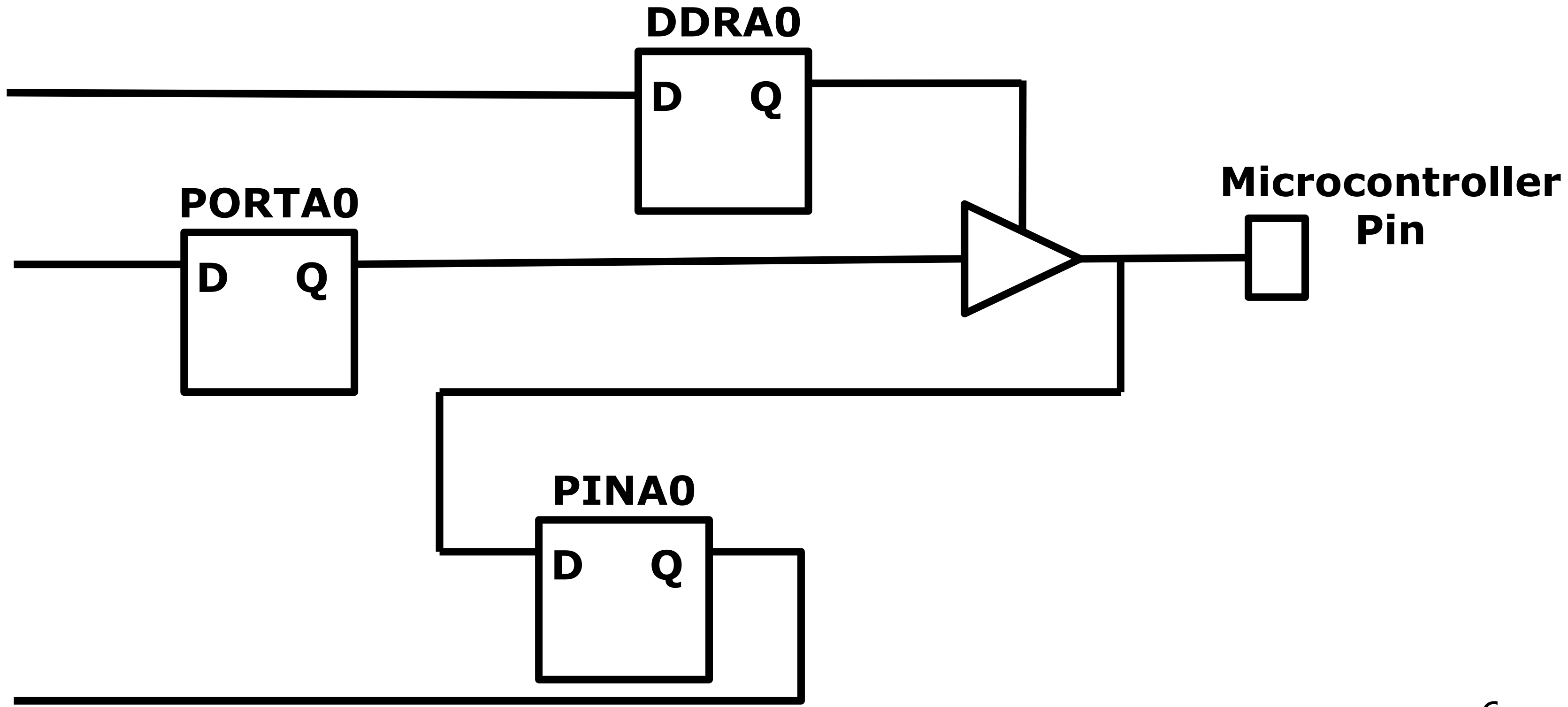
Extract from I/O Register List

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
0x18 (0x38)	TIFR3	-	-	ICF3	-	-	OCF3B	OCF3A	TOV3	144
0x17 (0x37)	TIFR2	-	-	-	-	-	OCF2B	OCF2A	TOV2	164
0x16 (0x36)	TIFR1	-	-	ICF1	-	-	OCF1B	OCF1A	TOV1	143
0x15 (0x35)	TIFR0	-	-	-	-	-	OCF0B	OCF0A	TOV0	114
0x14 (0x34)	Reserved	-	-	-	-	-	-	-	-	

...

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
0x0B (0x2B)	PORTD	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	98
0x0A (0x2A)	DDRD	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	98
0x09 (0x29)	PIND	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	98
0x08 (0x28)	PORTC	PORTC7	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0	98
0x07 (0x27)	DDRC	DDC7	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	98
0x06 (0x26)	PINC	PINC7	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0	98
0x05 (0x25)	PORTB	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	97
0x04 (0x24)	DDRB	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	97
0x03 (0x23)	PINB	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	98
0x02 (0x22)	PORTA	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	97
0x01 (0x21)	DDRA	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	97
0x00 (0x20)	PINA	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	97

PIN, PORT and DDR registers



Recall from lecture 12: *in* and *out* instructions

- **in *rd, P***
  - Load I/O register value into general purpose register
  - rd: r0 to r31
  - P = I/O register number (0 to 63, 0 to 0x3F) or name
  - Example:

- **out *P, rr***
  - Store value from general purpose register into I/O register
  - rr: r0 to r31, P = 0 to 63 (0 to 0x3F) or name
  - Example:

Which instructions will set all bits of port A to be outputs?

A.
```
ldi r16, 0xFF
out 0x02, r16
```
B.
```
ldi r16, 0xFF
out 0x01, r16
```
C.
```
ldi r16, 0x00
out 0x01, r16
```
D.
```
ldi r16, 0xFF
out 0x00, r16
```
E. None of the above

F. I don’t know

Using Port Names

- If we include **m324Adef.inc** file, we can use names for I/O registers rather than numbers:

  .include “m324Adef.inc”

  - Contents of this file are included directly in program
- Example - setting port A (all bits) to be an output:

Which instructions will set all bits of port B to be inputs?

1.
   ```
   ldi r16, 0xFF
   in DDRB, r16
   ```

2.
   ```
   ldi r16, 0x00
   in DDRB, r16
   ```

3.
   ```
   ldi r16, 0xFF
   in r16, DDRB
   ```

4.
   ```
   ldi r16, 0x00
   in r16, DDRB
   ```

5.
   ```
   ldi r16, 0xFF
   out DDRB, r16
   ```

6.
   ```
   ldi r16, 0x00
   out DDRB, r16
   ```

ser and clr instructions

- Short-hand ways of setting/clearing all bits of a register
- **`ser rd`** (**`ser`** = set register)
  - Exactly the same as **`ldi rd, 0xFF`**
    - Only works with registers >= 16
  - Example:

    `ser r16`
- **`clr rd`** (**`clr`**= clear register)
  - Exactly the same as **`eor rd, rd`**
  - Example:
    - **`clr r3`**

labels and jmp instruction

```
          jmp RESET
          …
RESET:    …
          …

loop:     …
          …
          jmp loop
```

AVR Assembly Language Program Template

```
    ; Comments start with semicolons
.include “m324Adef.inc”
    jmp RESET
    ; Interrupt handlers go here
    ; (more on this in week 9/10)

RESET:                  ; Starting point
    ...                 ; Initialisation code
                        ; e.g. set DDRs
loop:
    ...                 ; Do stuff
    jmp loop            ; Repeat
```

Task 1 - AVR Assembly Program

- Write an AVR program to
  - read the value on port A pins
  - flip (invert) all the bits
  - write the value out to port C
  - do this repeatedly
- Remember to set data direction registers appropriately before reading/writing values
- Enter the code into Microchip Studio, simulate your code, check that it works by checking the appropriate registers

```
.include "m324Adef.inc"
    jmp RESET

RESET:                  ; Starting point
    ...                 ; Initialisation code
                        ; e.g. set DDRs
loop:
    ...                 ; Do stuff
    jmp loop            ; Repeat
```

Task 2 – Programming the AVR Board

- Connect 8 switches on the IO Board to AVR port A; and 8 LEDs to AVR port C
  - Use 4-way cables with 4-way headers on each end
  - Build and download the program you wrote in task 1
- See the Pololu Programmer Guide on Blackboard
- Test your program – change the values on the switches connected to port A and observe the LEDs
- Modify your program to output the two's complement (negative) of the input, rather than the ones' complement (inversion). Build, download and test it.

Task 3 – Additional task

- Write an AVR assembly language program to repeatedly
  - Read the value on port A pins (8 bits)
  - Read the value on port D pins (8 bits)
    - Set the upper 4 bits of this value (read from port D) to 0 (consider bitwise AND to do this)
- Add the two values and output the result to port C
- Make sure you set the data direction registers appropriately.
- Connect 8 switches on the IO Board to AVR port A; and 8 LEDs to AVR port C; and connect the 4 push buttons on the IO Board to AVR port D (4 least significant bits)
- Use individual jumper wires for this if required
- Build, download and test your program and verify it behaves as expected