CSC230 Midterm Review

Read Ch3, Ch9, pg411-424, pg275-287

- Kilo -> mega -> giga
- Each processor has their own machine language
- Control Unit (CU), ALU, registers in CPU
- CU fetches, decode, and executes instructions
- Program counter keeps track of the address of the next instruction to be fetched
- As instructions are fetched, the PC is updated to the next instruction by incrementing by 1

- CISC vs RISC processors
- RISC is easy to implement in hardware
- CISC varies from simple to complex instructions

- **Princeton architecture** – stores program instructions and data in the same memory
- Instructions and data travel from memory in the processor using the same address and data bus
- **Harvard architecture** – stores instructions and data separately
- They have different data pathways
- Allows parallel access to instructions and data
- Can fetch and decode next instruction same time as current instruction is being read/written data in memory

- System bus contains data bus, address bus, control bus
- The bus transfers information between unit
- Address = location, data = info, control = read/write

- Primary storage: RAM is volatile, expensive, fast, and small
- Secondary storage: ROM, Hard drives, non-volatile, slow, big, cheap

Execution Cycle

CPU

1. Fetches instructions

   Address in PC -> MAR
   MAR -> address bus
   Read signal instruction sent by the control unit -> control bus to memory
   Wait for memory to read content location
   Content location -> data bus
   Data bus -> MDR
   MDR -> IR

2. Update current pointer to instructions memory address

Increment PC using ALU

3. Decode instruction

    Decoder is located inside control unit
    Looks at the opcode and operands

4. Fetch further

    MAR, MDR, PC could be in use

5. Execute

    Uses the ALU
    Read/write to memory using CU

6. Step 1

Assembly in AVR

- AVR uses RISC Microcontroller
- AVR instructions are 16 bits
- Program memory is 16 bit words
- Program memory are word addressable (address 0 =16 bits, address 1 = 16 bits etc)
- AVR has 32 8 bit registers (R0-R31)
- Data that is currently being manipulated must be in these registers
- Operands – the data, or address of data to be used
- Opcode – implicit operand (the code that tells if it is add, ldi, or mov etc)

NOTE: All 4 instructions access memory ONCE to fetch the instruction itself (mov, ldi, add, st) but only ST has one more instruction that accesses the memory again during the execution step.

- The processor knows how many bits of opcode is used in the current instruction because the Opcode in AVR begins left most bit of the word. For example “add” starts with 0b0000 11; there cant be any other opcode that use the same bits and binary numbers.
- AVR uses 2 stage pipeline of Harvard architecture, can exec instructions while fetching next (EXECUTION TRACE)
- Jump is relative jump: jumps to a location relative to the current PC. Rjmp K where K is 12 bits varies from -2048 to 2048.

- AVR uses Flash, SRAM and EEPROM memory.
- Flash memory used for program: stores 16 bits (non-volatile)
- SRAM memory used for data: stores 8 bits (volatile)

- Directives:

    .db .dw .dd .dq – defines series of byte, word, double word, quad word **(for FLASH)**
    .byte – reserve an address size **(for SRAM)**
    .def – define a name for a register **(for SRAM)**
    .equ or .set – set a symbol to a value or expression **(for SRAM)**

.cseg .dseg .eseg – switches the memory segment (flash, sram, essprom)
.org – set the location counter of the current memory segment to a value

.dw 0x3412 will store in flash memory (.cseg) as [12][34] because AVR uses little endian.

Functions used in expressions

- LOW returns low byte of an expression
- HIGH returns second byte of an expression
- BYTE2 same as HIGH
- BYTE3 returns third byte of an expression

Data memory Map for 2560

- Internal SRAM memory starts at **0x200 to 0x21FF**
- External SRAM: 2200 to FFFF
- 32 registers: 0-1F
- 64 I/O registers: 20-5F
- External I/O registers: 60-1FF

X pointer: R26;27
Y pointer: R28;29
Z pointer: R30;R31

DIFFERENCES BETWEEN ABSOLUTE, RELATIVE, INDIRECT*
LDD
LDS
STD
STS
LDI
LPM
RJMP
JMP
RCALL
CALL
SBR
SBI
SBIS
CBR
CBI
SBIC

- Immediate addressing: instructions contain the value of the operand
- Register direct: Single register (ex: INC R16, CLR R22)
- Register direct: Two registers (ex: ADD R16, R17)
- I/O Direct addressing (ex: IN R16, PIND)
- Direct Data addressing (ex: STS 0x1000, R16)
- Indirect Data addressing (ex: LD R16, Y)

- Indirect Data w/Displacement (ex: LDD R16, Y+0x10)
- Indirect Data: Pre-decrement (ex: LD R16, -Z)
- Indirect Data: Post-decrement (ex: LD R16, Z+)
- Program memory addressing (ex: LPM)
- Indirect Program addressing (ex: IJMP, ICALL)
- Relative program addressing (ex: RJMP, RCALL)

Ports in AVR

- Port mapped I/O can only use IN and OUT
- Port mapped uses separate buses for CPU to memory, and CPU to peripherals
- Has special instructions for accessing ports

- Memory mapped I/O can only use load and store
- Memory mapped has a single bus for CPU to memory elements
- The I/O are connected to the single memory and are treated like memory
- CPU read/write data to locations and are later defined as I/O devices or as memory
- Disadvantage is that there is less space in memory for other things
- Has no extra instructions

- Each port has 8 pins
- Each pin can be set as input or output
- Pins that are set to output use PORTx
- Pins that are set to input use PINx
- DDRx initializes a port

Arithmetic and Logic

- AVR is a little endian machine; the address will be stored reversed. (E00A -> [0A][E0])
- 8 bit containers can have 2^8 = 256 distinct values
- Half adder (takes 2 bits A and B)
- Full adder (takes 3 bits A, B and C)

Carry Flag (C) – when the register overflows for arithmetic or logical operations

Carry flag if there is a carry past the 8th bit

Ex) 1111 + 0001 = 1| 0000

Zero Flag (Z) – if the result of the bits is equal to 0 (ex: 0001 + 1111 = 1|0000)

(V) – Overflow Flag – (0|111 + 1|000)

Overflow if left most bit of the 2 values is the same and result is the opposite, there is an overflow

Ex) 0100 + 0100 = 1000

Negative (N) – If the most significant bit is 1