The Microprocessor

Dr Geoff Merrett
ELEC1202: Digital Systems and Microprocessors

This Lecture

• The Rise of the Microprocessor
  – Moore’s Law
  – Microprocessors vs Microcontrollers

• From Program to Machine
  – Instruction sets, Assembler and Machine code

• Basic Computer Architecture
  – The Fetch-Execute Cycle
Early Microprocessors

• Intel 4004, in 1971
  – The inventors designed the 4004 while building a custom chip for a Japanese calculator maker.
  – 2,250 transistors, 10um process
    • *Intel i7: 1,170,000,000 transistors, 32nm process!*
  – 4-bit Arithmetic-Logic Unit, maximum clock speed of 108kHz, yielding just over 50,000 operations per second.
    • *Today: few GHz clock*
  – 640 bytes of memory

• Intel 8008, 1972
  – 8-bit version of the 4004
  – 3500 transistors
  – The 8008 had 14 address lines and hence could address 16Kbytes of memory
16-Bit Microprocessors

- Motorola 68000, 1978
- Intel 8086 and 8088, 1978
- Motorola 68020 (internally 32-bit), 1984
- Intel 80286 (OS support), 1982

- **IBM PC**: why Intel won the ‘war’ with Motorola.
The BBC Micro

- Released in 1981
- 2MHz CPU
- 32K Memory
The Rise of the Microprocessor

Computational intensity

- 1950: Military code decryption, navigational systems
- 1960: Rocket control systems
- 1970: CAD, VLSI simulations
- 1980: Particle and cosmology models
- 1990: Mass communication
- 2000: First microprocessor 1971

MICROPROCESSOR ERA

Time
Moore’s Law

- What is Moore’s Law?
  - Is it the Transistor count? Transistor density? Performance?
  - Is it supposed to double every 12 months, 18 months, 24 months?

- In 1965, Gordon Moore (Intel Co-founder) predicted that:
  - Transistors on an IC would double every year for at least the next ten years.
  - Specifically, “the number of components per IC for minimum cost” would double
Moore’s Law

- What is Moore’s Law?
  - Is it the Transistor count? Transistor density? Performance?
  - Is it supposed to double every 12 months, 18 months, 24 months?

- In 1965, Moore predicted that transistors on an IC would double every year.

- In 1975, he updated this:
  - Transistors on an IC would double every two years.

Moore’s Law

- What is Moore’s Law?
  - Is it the Transistor count? Transistor density? Performance?
  - Is it supposed to double every 12 months, 18 months, 24 months?

- In 1965, Moore predicted that transistors on an IC would double every year.

- In 1975, he updated this:
  - Transistors on an IC would double every two years.

- David House (Intel) factored in the increasing performance of transistors:
  - the performance of ICs would double every 18 months.
Moore’s Law

• Has Moore’s Law been correct?
• Will Moore’s Law last forever?

• In 2005, Gordon Moore said that:

  "It can't continue forever. The nature of exponentials is that you push them out and eventually disaster happens."

• Why?
  – Reducing transistor sizes are begin to reach the size of atoms
  – Properties of light make fabricating ICs using photolithography unsuitable.

• But...
  – Moore’s law has effectively ‘driven’ the pace of development in integrated circuits
  – Today, extends beyond simply ‘transistor density’ today
  – Many barriers have been broken over the past 40+ years

Microprocessor or Microcontroller?

• A microcontroller is a complete computer on a single chip, i.e. a microprocessor with program and data memory, and peripherals – eg:

  - Interfaces
    - I2C/SPI
    - UART
    - USB
    - Ethernet

  - Input/Output
    - Digital I/O
    - ADC/DAC

  - Other
    - Timer
    - Real Time Clock (RTC)
    - LCD Controller
    - OpAmp/Comparator
    - Camera Controller
    - Radio Transceiver

Microprocessor or Microcontroller?

- **Average home:**
  - 40-50 microprocessors.

- **7-Series BMW/Mercedes S-Class**
  - 100 microcontrollers

- **Land Rover Evoque**
  - 80 control units
  - ~100 million lines of code
Atmel ATmega644

Source: Atmel / www.atmel.com
Atmel ATmega2560

Source: Atmel / www.atmel.com
What about the BBC Micro?

- Sold ~1.5 million units

- Succeeded by the Acorn Archimedes
  - Custom 32-bit RISC microprocessor – ARM2
  - “Acorn RISC Machine”, or ARM
  - ARM2: 8MHz (giving 4MIPS)
  - Intel 80386: 16MHz (giving 5MIPS)

- 1990, Research division spun off as “Advanced RISC Machines” (ARM!)
  - Don’t compete for performance alone, but do for energy-efficient performance
  - Now: FTSE 100 company, turnover >£700M (2013)
  - >50 billion chips sold (10 billion in 2013)
  - >95% of smartphone market (2010)
Introduction to Computer Architecture

- Flash Program Memory
- Instruction Register
- Instruction Decoder
- Program Counter
- General Purpose Registers
- Arithmetic Logic Unit (ALU)
- Data (SRAM)
From Program to Machine

- A reasonably simplistic view:

High-Level Programming Language (e.g. C) → Assembly Language → Machine Code
AVR Instructions

- Each AVR program instruction has 16 bits
- An AVR instruction in assembler format:

```
dec r1     ;decrement register 1
```

- Machine code:

```
0x941A -> 1001 0100 0001 1010
```

16-bit machine code

*Each AVR instruction has 16 bits*
AVR Instructions

• An AVR instruction in assembler format:

```
mov r3, r4     ; move contents of r3 to r4
```

- **Mnemonic**: `mov`
- **Source operand**: `r3`
- **Destination operand**: `r4`

• Machine code:

```
0010110001000011
```

- **Opcode for 'mov'**: `0x2C83`
- **Source register**: `001011` (r3)
- **Destination register**: `00100` (r4)
AVR Instructions

**ALU Instructions**
- **ADD**: Add two registers
- **ADC**: Add two registers (with carry)
- **ADIW**: 16bit Add Immediate
- **AUB**: Subtract two registers
- **AND**: Logical AND
- **OR**: Logical OR
- **EOR**: Exclusive OR
- **COM**: One’s Complement
- **NEG**: Two’s Complement
- **INC/DEC**: Increment/Decrement
- **SER/CLR**: Set/Clear Registers
- **MUL**: Multiply
- **DES**: Apply DES Data Encryption

**Data Transfer Instructions**
- **MOV**: Move data between registers
- **LD/LDI/LDS/LDD**: Load to a register
- **ST/STS/STD**: Store from a register
- **IN/OUT**: In and Out Ports
- **PUSH/POP**: On and off stack
- **LPM/SPM**: Load/store program mem.

**Branch Instructions**
- **JMP/CALL**: Direct jump
- **IJMP/ICALL**: Indirect jump
- **RJMP/RCALL**: Relative jump
- **RET/RETI**: Return from call/interrupt
- **CP**: Compare
- **SB**: Skip if bit in Register or I/O is set/clr
- **BR**: Branch if condition is met
The AVR Instruction Set

- For more information on the AVR Instruction Set:
From Program to Machine

```c
int main(void)
{
    int i = 3, j = 5;
    i = i + j;
    if (i > j)
        j = 0;
}
```

Some of this lecture was given using the whiteboard/interactive demonstrations.
Refer to the notes you made during the lecture
The Fetch-Execute Cycle

- The ‘instruction cycle’ which runs programs on a computer.
- A simplified fetch-execute cycle:
  - Inspect the Program Counter to find out the memory address of the next instruction
  - Fetch the instruction at that address, and put it in the Instruction Register
  - Increment the Program Counter so that it points to the next address
  - Decode the instruction in the Instruction Register, and setup control signals
  - Execute the instruction
int main(void)
{
    int i = 3, j = 5;
    i = i + j;
    if (i > j)
        j = 0;
}

(ARM Assembler) – see http://assembly.ynh.io/
The Fetch-Execute Cycle

• The ‘instruction cycle’ which runs programs on a computer.

• A simplified fetch-execute cycle:
  – Inspect the Program Counter to find out the memory address of the next instruction
  – Fetch the instruction at that address, and put it in the Instruction Register
  – Increment the Program Counter so that it points to the next address
  – Decode the instruction in the Instruction Register, and setup control signals
  – Execute the instruction

*NOTE: this is a somewhat artificial example, as it’s an AVR microcontroller and ARM instructions...
The majority of this lecture was given using the board/visualiser.

Refer to the notes you made during the lecture.
Questions?