Covered in Lecture 5:
- Quick Introduction to Microcontroller Subsystems
- Microcontroller Registers
- Microcontroller Modes (Single chip, Extended, etc..)
- EVBU Memory Maps

Covered in this section:
M68CH11 CPU
(Note: the Central Processing Unit (CPU) is the “core” of the microcontroller where instructions are executed)
The 68HC11 CPU contains:

- Circuits to process instructions
- CPU Registers

(Note: M68HC11 CPU registers are an integral part of the CPU and are not addressed as if they were memory locations)

Figure 1-2. M68CH11 Programmer’s Model (Reference)
MC68HC11 Accumulators A, B, and D

• A & B are:
  – 8-bit registers
  – Can be used for 8-bit math operations (Note: This is why A & B are called “Accumulators”)
  – Can also be used for 8-bit binary logic, comparisons, memory transfers, etc…

• D is:
  – 16-bit register
  – Cannot be used when A or B is in use
  – Can be used for 16-bit math in conjunction with Index X & Y
  – Can be used for 16-bit memory transfers, comparisons, etc..

MC68HC11 Index Registers X & Y

• Can be used for 16-bit math with Accumulator D
• Mainly used for addressing memory in Indexed mode (Note: Indexed addressing mode will be covered in a later section)

MC68HC11 Program Counter (PC)

• Contains the address of the next instruction to be executed
An interrupt can be recognized at any time after it is enabled by its local mask, if any, and by the global mask bit in the CCR. Once an interrupt source is recognized, the CPU responds at the completion of the instruction being executed.

Interrupt latency varies according to the number of cycles required to complete the current instruction. When the CPU begins to service an interrupt, the contents of the CPU registers are pushed onto the stack in the order shown in Table 5-5. After the CCR value is stacked, the I bit and the X bit, if XIRQ is pending, are set to inhibit further interrupts. The interrupt vector for the highest priority pending source is fetched and execution continues at the address specified by the vector. At the end of the interrupt service routine, the return-from-interrupt instruction is executed and the saved registers are pulled from the stack in reverse order so that normal program execution can resume.

<table>
<thead>
<tr>
<th>Memory Location</th>
<th>CPU Registers</th>
</tr>
</thead>
<tbody>
<tr>
<td>SP</td>
<td>PCL</td>
</tr>
<tr>
<td>SP–1</td>
<td>PCH</td>
</tr>
<tr>
<td>SP–2</td>
<td>IYL</td>
</tr>
<tr>
<td>SP–3</td>
<td>IYH</td>
</tr>
<tr>
<td>SP–4</td>
<td>IXL</td>
</tr>
<tr>
<td>SP–5</td>
<td>IXH</td>
</tr>
<tr>
<td>SP–6</td>
<td>ACCA</td>
</tr>
<tr>
<td>SP–7</td>
<td>ACCB</td>
</tr>
<tr>
<td>SP–8</td>
<td>CCR</td>
</tr>
</tbody>
</table>

Table 5-5. Stacking Order on Entry to Interrupts (Technical)
RTI, RETURN FROM INTERRUPT

INTEGRUPT ROUTINE
PC $3B = RTI

7 STACK 0
SP
SP+1 CCR
SP+2 ACCB
SP+3 ACCA
SP+4 IXH
SP+5 IXL
SP+6 IYH
SP+7 IYL
SP+8 RTNH
SP+9 RTNL

LEGEND:

- RTN = ADDRESS OF NEXT INSTRUCTION IN MAIN PROGRAM TO BE EXECUTED UPON RETURN FROM SUBROUTINE
- RTN H = MOST SIGNIFICANT BYTE OF RETURN ADDRESS
- RTN L = LEAST SIGNIFICANT BYTE OF RETURN ADDRESS
- ↑ = STACK POINTER POSITION AFTER OPERATION IS COMPLETE

SWI, SOFTWARE INTERRUPT

MAIN PROGRAM
PC $3F = SWI

↑ SP-9 CCR
SP-8 ACCB
SP-7 ACCA
SP-6 IXH
SP-5 IXL
SP-4 IYH
SP-3 IYL
SP-2 RTNH
SP-1 RTNL

WAI, WAIT FOR INTERRUPT

MAIN PROGRAM
PC $3E = WAI

↑ SP
SP RTNL

George W. Woodruff School of Mechanical Engineering, Georgia Tech
MC68HC11 CONDITION CODE REGISTER

**MASKING BITS**

- **S** – Disables STOP instruction when set.
- **X** – Masks XIRQ Request when set.
  - set by hardware reset, cleared by software
  - set by unmasked XIRQ
- **I** – Masks interrupt request from all IRQ level sources (both external and internal) when set.
  - set by unmasked I level request or unmasked XIRQ
- **V** – 2’s complement overflow indication
- **Z** – Zero result
- **N** – Negative (follows MSBit of result)
- **H** – Half Carry from bit 3 to bit 4
  - ADD operations only

**ARITHMETIC BITS**

- **C** – Carry/Borrow from MSB unsigned arithmetic
- **V** – 2’s complement overflow indication
  - signed arithmetic
- **Z** – Zero result
- **N** – Negative (follows MSBit of result)
- **H** – Half Carry from bit 3 to bit 4
  - ADD operations only

```
7 01001111
  +00001000
    01010111
```

H will be set to 1
Example 3 from Lecture 4:

- $85_{10} - 90_{10} = -175_{10}$

$85_{10} = 55_{16} = 0101 0101_2$
$1010 1010 = 1's$ comp. of $55_{16}$
$1010 1011 = 2's$ comp. of $55_{16}$

$90_{10} = 5A_{16} = 0101 1010_2$
$1010 0101 = 1's$ comp. of $5A_{16}$
$1010 0110 = 2's$ comp. of $5A_{16}$

$1010 1011 = (2's$ comp. of $55_{16})$
$1010 0110 = (2's$ comp. of $5A_{16})$

--------
$1 0101 0001 = +51_{16} = +81_{10}$

V bit will be set.
C bit will be set.
QUESTIONS???