EE5314 Class Notes

33FJ128MC802 Microcontroller Architecture

Spring 2011

Dr. Jason Losh

Architecture Topics

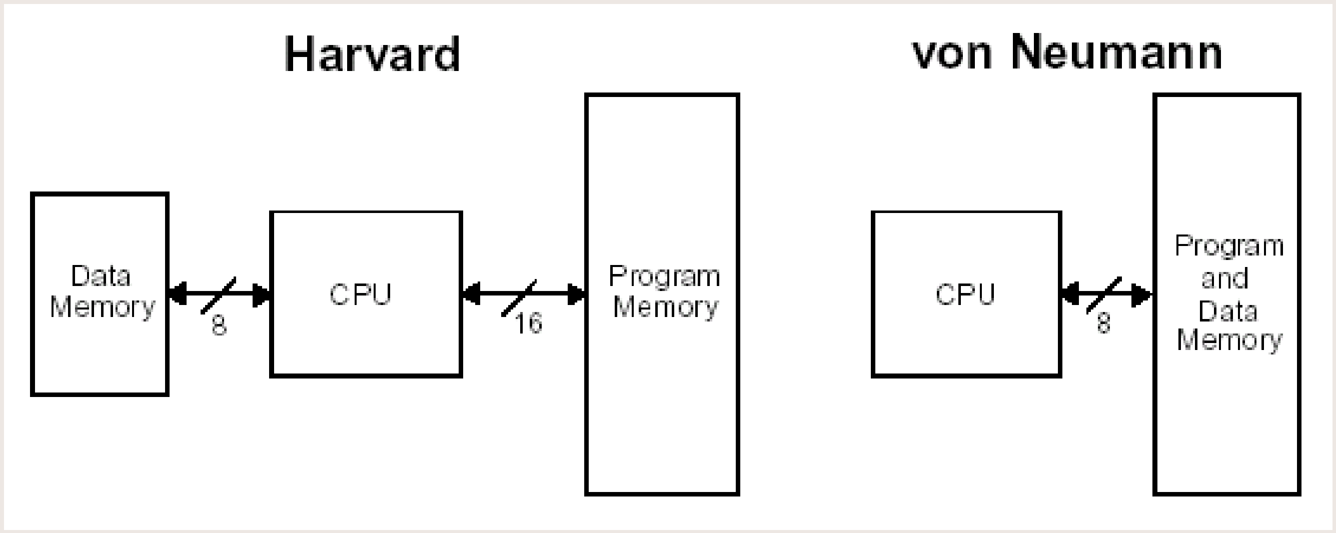
- Harvard v. von Neumann
- RISC v. CISC
- Microcontrollers v. Microprocessors
- PIC Microcontroller Architecture
  - Pipelines
  - ALU and CPU
  - Data memory
  - Program memory
  - Reset and interrupt operation
  - Peripherals

Resources

- Resources on the Class Server
  - Microchip 33FJ128MC802 Datasheet
  - Microchip 33F Family Reference Manual
  - Microchip 16-bit MCU and DSC Programmer’s Reference Manual

Harvard v. von Neumann

- von Neumann Architecture shares a memory space for programs and data
- Harvard Architecture has different memory spaces for program and data



From Microchip 18C Family Reference

Harvard v. von Neumann

- von Neumann Architecture
  - Shares a memory space for programs and data
  - Programs can be altered or loaded easily
  - Code storage may not be optimal and may require multiple fetches to form an instruction
  - Contention exists between program fetches and data flow, causing a performance hit
- Harvard Architecture
  - Has different memory spaces for program and data
  - Allows different size buses for program and data memories
  - Allows optimal sizing of program data bus width to facilitate fast execution of instructions

RISC v. CISC

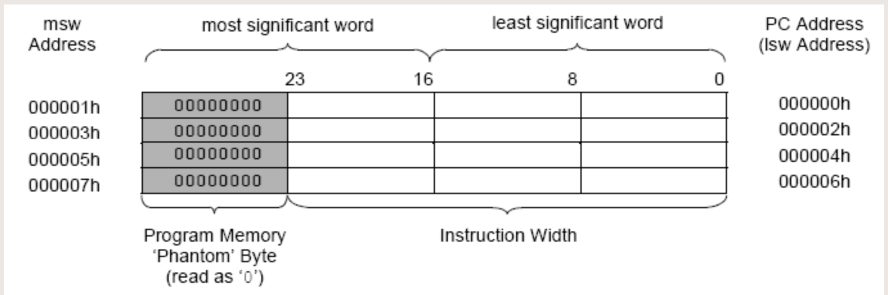
- CISC (Complex instruction set computer)
  - Commonly many hundreds of instructions
  - Specialized instructions perform more complex functions
  - Complicated instructions can take 50+ cycles (*, /)
- RISC (Reduced instruction set computer)
  - Generally less than 100 instructions
  - Instructions are less powerful on average, so many instructions may be needed to perform a CISC equivalent instruction
  - Instructions normally execute in 1 instruction cycle leading to higher MIPS performance

Microcontrollers v. uP’s

- Microprocessors
  - Provide external data and address buses
  - Generally require external memory
  - Require many external peripherals for typical applications
- Microcontrollers
  - Generally do not expose data and address buses
  - Generally do not require external memories
  - Have many peripherals built-in

33FJ128MC802

- Modified Harvard architecture
  - 16MB program memory space organized as 4M x 32-bits (23-bit word address, 24-bit data bus) with 44032 x 24-bits of flash populated



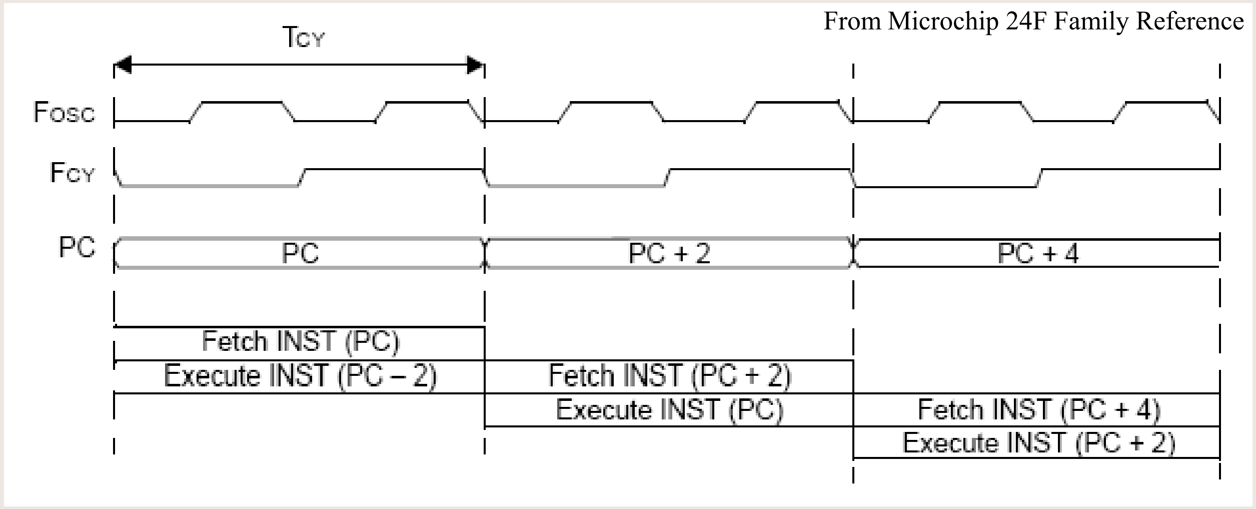
  - 64kB data memory space organized as 32k x 16-bits (16-bit byte address, 16-bit data bus) with 16kB of SRAM populated
  - MODIFIED: Part of program space is visible in the data space

33FJ128MC802

- RISC-based
  - 83 base instructions, most execute in 1 instruction cycle
- Microcontroller with DSP core
  - DC to 80 MHz clock (3-3.6V,-40 to 85/125ºC)
  - 2 clks / instruction cycle ($T_{cy} = 2 / f_{clock}$), upwardly bounded to a 40 MIPS limit
  - 28-pin device with 21 pins available for i/o
  - Internal A/D, PWM, timers, CAN, serial, SPI, RTCC, QEI, DMA, …

Pipeline Operation

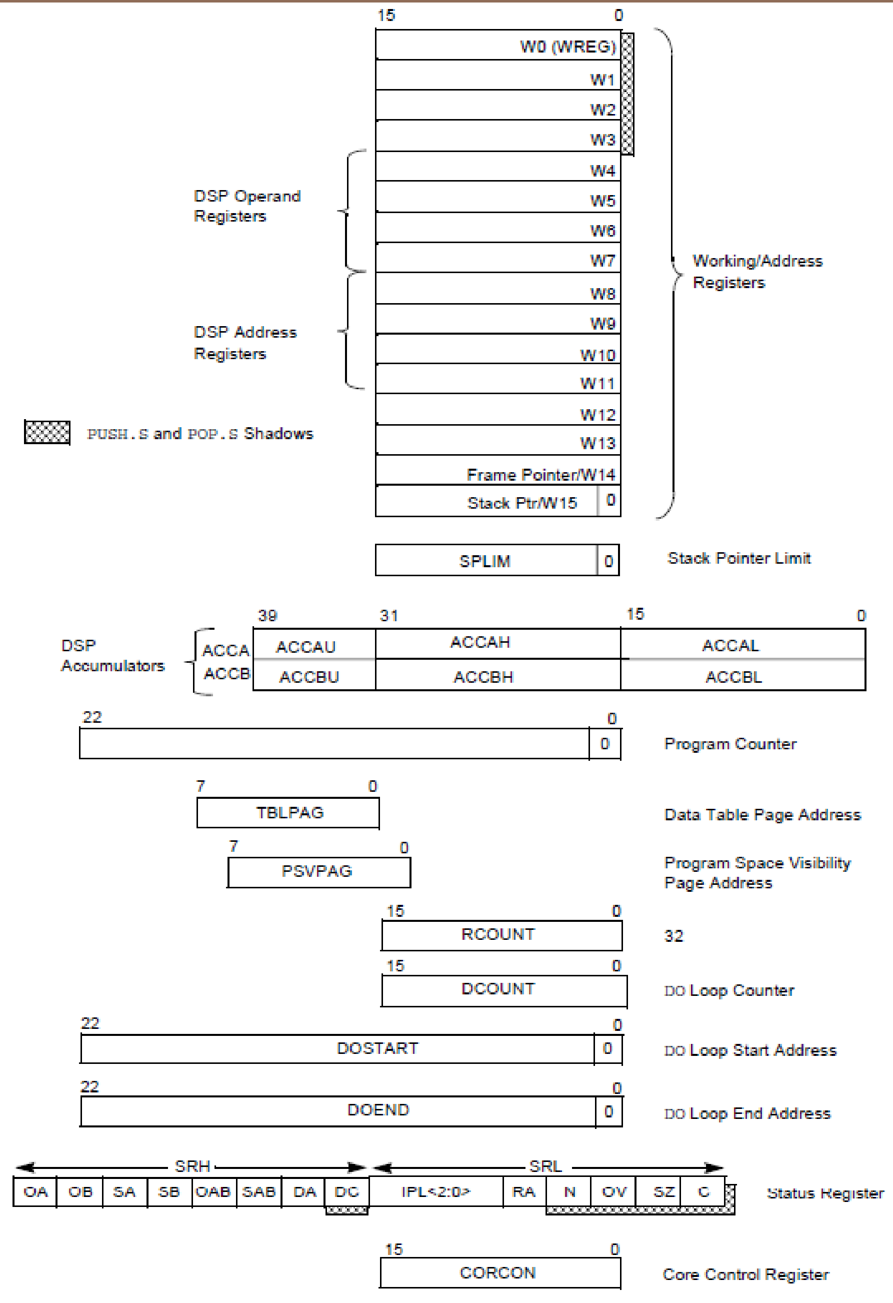
- The 24FJ64GA002 use a simplified 2 stage pipeline (fetch/execute)
- Some double word commands and discontinuities in the PC (branches, calls, returns) result in a 1 or 2 clock penalty



8086 Registers (for comparison)

- 20b Program counter (CS:IP)
- 20b Stack pointer (SS:SP, SS:BP)
- 20b Indirection regs (DS/ES:BX+DI/SI)
- 16b General regs (AX, BX, CX, DX)
- 16b Repeat Counter (CX)
- 16b Port register (DX)
- 16b Status register (FLAGS)

Registers



Note: DCOUNT, DOSTART and DOEND have one level of shadow registers (not shown) for nested DO loops.

From Microchip 33F Family Reference

Working and Status Registers

- W0-W14 are 16b working registers
  - Similar to general use of AX, BX, CX, DX
  - W0 (WREG) is similar to AX (accumulator)
  - Can be used as indirect memory pointers (like DI, SI, BP, BX)
  - W0-3 are shadowed for quick storage and recall
- Status registers
  - Z, C, DC, OV, and N flags
  - CORCON<3>:IPL (Processor interrupt priority)
  - RA/DA (Repeat/do loop active flags)
  - Ox/Sx (DSP overflow and saturation flags)

Pointer Registers

- W15, SPLIM, and W14 are the 16b stack pointer and stack pointer limit (and frame pointer)
- PSVPAG is the 8b program visibility register that allows a portion of the program memory to be seen in the data memory space
- TBLPAG is the 8b table pointer upper byte that allows a constant table to be accessed in the program memory

Flow Control Registers

- PC is the 23b program counter
- RCOUNT is 16b repeat counter for one instruction
- DCOUNT (16b counter), DOSTART, and DOEND (23b address pointers) add DO loop functionality and are shadowed to allow nested DO loops

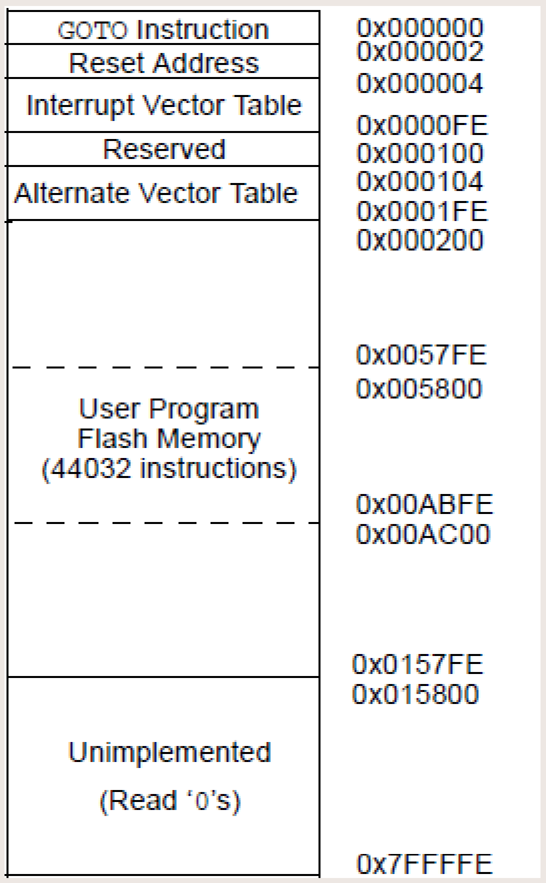
Control Registers

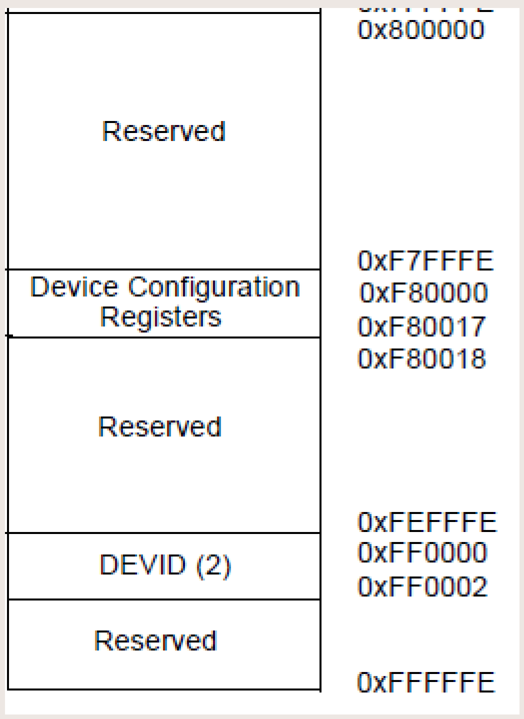
- DISICON is used to disable lower-level interrupts (1-6) for a number of clock cycles
- CORCON is a general control register for the CPU and also contains DSP configuration bits
- MODCON is used for the DSP modulo configuration

DSP-Related Registers

- ACCA and ACCB are 40b accumulators for the DSP which can be sub-addressed as words (ACCxL, ACCxH, and ACCxU)
- CORCON controls saturation modes
- MODCON, XMODSRT, XMODEND, YMODSRT, and YMODEND for circular buffers (modulo addressing)
- XBREV sets the buffer size for bit reverse order addressing, like in FFTs

Program Memory





From Microchip 33FJ128MC802 Datasheet

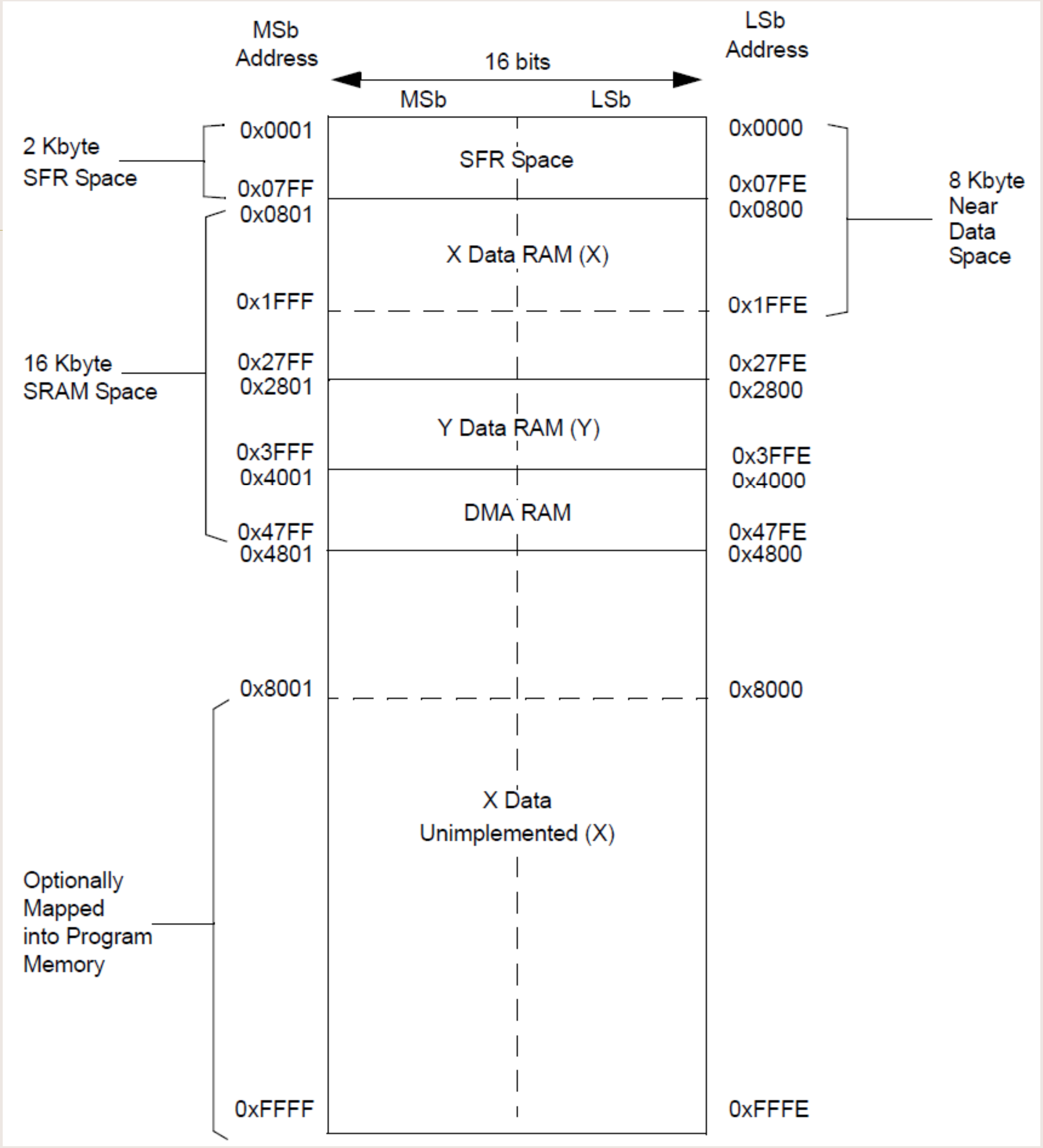
Program Memory

- 0x000000-0x000003
  - The processor jumps to address 0 on reset
  - A GOTO *add* instruction (4 words long) exists at this address
- 0x000004-0x0001FF
  - Interrupt Vector Tables exist in this range
- 0x000200-0x0157FF
  - Flash memory for program storage

Program Memory

- 0x800000-0x800017
  - Non-volatile (flash) copy of the device configuration registers
  - These contain information on how the device is configured to operate
- 0xFF0000-0xFF0003
  - Used to identify the silicon revision and part

Data Memory


MSb Address
16 bits
LSb Address
MSb
LSb
0x0001
0x07FF
0x0801
0x1FFF
0x27FF
0x2801
0x3FFF
0x4001
0x47FF
0x4801
0x8001
0xFFFF
0x0000
0x07FE
0x0800
0x1FFE
0x27FE
0x2800
0x3FFE
0x4000
0x47FE
0x4800
0x8000
0xFFFE
2 Kbyte SFR Space
16 Kbyte SRAM Space
Optionally Mapped into Program Memory
8 Kbyte Near Data Space
SFR Space
X Data RAM (X)
Y Data RAM (Y)
DMA RAM
X Data Unimplemented (X)

Data Memory

- 0x0000-0x07FF
  - Special function register space
  - Contains core registers
  - Contains registers to control peripherals
- 0x0800-0x47FF
  - Populated with 16 kB SRAM
- 0x8000-0xFFFF
  - Can be mapped to a range of the Program Memory using the PSVPAG register

DSP and DMA Data

- The data from 0x0800-0x47FF is split into two spaces (X and Y) which allow 2 accesses at once by using two address generation units (AGUs)
  - The general controller instructions treat as a unified space
  - The DSP can access an address in X and Y with the same instruction (i.e. FIR filters)
  - The upper 4 kB of Y can also be used for DMA operations in the background

Program Space Visibility

- Allows a 32kB range of program memory to be accessible in the upper half of the data memory space
- This is a modified Harvard architecture
- The configuration space (program memory $\geq$ 0x80000) not accessable with PSV
- Only the LSW of a 24b instruction word can be accessed

Program Space Visibility

- PSV is turned-on by asserting the PSV bit (CORCON<2>)
- If the effective address (EA) ≥ 0x8000, then
  - The 24b program word address is: 0:PSVPAG<7:0>:EA<14:1>:0
  - Since data memory accesses bytes, the even/odd part of the program word is selected by EA<0>

Program Space Visibility Example

- Consider
  - MOV #0, W0
  - MOV W0, PSVPAG
  - MOV #0x8294, W1
  - MOV [W1], W0
- This moves the word at EA = 8294h = 0b1**000 0010 1001 010**0 into the W0 register
- Since EA<15> = 1, this is accessing program memory
- 24b program word address is 0xb0000 0000 0**000 0010 1001 010**0 = 0x00294
- Word at this address is loaded into W0

Program Space Visibility Example

- Consider
  - MOV #0, W0
  - MOV W0, PSVPAG
  - MOV #0x8295, W1
  - MOV.B [W1], W0
- This moves the word at EA = 8295h = 0b1**000 0010 1001 010**1 into the W0 register
- Since EA<15> = 1, this is accessing program memory
- 24b program word address is 0xb0000 0000 0**000 0010 1001 010**0 = 0x00294
- MSB of LSW at this address is loaded into W0

Data Storage

- Data can be accessed from any byte or from an even word (misaligned words cause an exception)
- Near Data Space
  - The lower 8kB of data memory (0x0000-0x1FFFF) is prized
  - A group of instructions use a limited 13b coding of the data address to allow compact (2 word) instruction size

Code Storage and PC

- PC is set to 0 on reset
  (GOTO main should be stored at add 0)
- Instructions are 2 words or 4 words in length
- Instructions start on 2 word boundaries (4 byte boundaries)
- PC is word counter, which increments by 2 and thus advances by 4 bytes in the space

Stack Operation

- Stack Pointer (W15)
  - W15 is initialized to 0x800 on reset
  - W15<0> is always 0 to force word alignment
  - Push is little-endian format, post inc by 2
- Stack Exceptions
  - If W15 < 0x800, then a stack underflow exception is raised
  - If W15 > SPLIM, then a stack overflow exception is raised

Stack Operation

- Frame Pointer (W14)
  - Used with LNK and ULNK instructions in subroutines to allocate local variables on the stack and to ease variable passing to subroutines
- PUSH Wn RTL is:
  [W15] ←Wn, W15+=2
- POP Wn RTL is:
  W15-= 2, Wn ← [W15]

Stack Operation

- CALL *add* RTL is:
  [W15] ←LSW (*PC)*, W15+=2
  [W15] ←0x00:MSB (*PC)*, W15+=2
- Interrupt invocation RTL is:
  [W15] ←*PC<15:0>*, W15+=2
  [W15] ←SRL:CORCON<3>:*PC<22:16>*, W15+=2
  (SRL contains all but the DC flag SR<8> of the status register, so usually this is sufficient to save the context)

Data Instructions

- Commands read from left to right
- Example: MOV #lit16, Wd
  - #lit16 is a 16b constant, stored as part of the instruction
  - Wnd is the destination working register n
  - MOV #0x1234, W2 RTL is W2 ← 1234h

Data Addressing Modes

- Register to Register Mode
  - 8086: MOV AX, BX or MOV AL, BL
  - 33FJ: MOV Ws, Wd or MOV.B Ws, Wd
- Immediate Mode (Literal)
  - 8086: MOV AX, 1234h or MOV AL, 12h
  - 33FJ: MOV #0x1234, Wd
    or MOV.B #0x12, Wd
- Direct or Extended Mode
  - 8086: MOV AX, [1000h]
  - 33FJ: MOV 0x1000, Wd

Data Addressing Modes

- Indirect Mode
  - 8086: MOV AX, [DI], MOV AX, [BX+DI]
  - 33FJ: MOV [Ws], Wd, MOV [Ws+Wb], Wd
- Constant Table Read/Write
  - TBLRDL/TBLWTL accesses the LSW of a program word
  - TBLRDH/TBLWTH accesses the MSW of a program word

DSP Core and Addressing

- 40-bit saturating accumulators
- 40-bit barrel shifter
- 17x17 bit multiplier
- Can read from two memory addresses per instruction cycle (one in X space and one in Y space)
- Supports linear and circular addressing
- Supports bit-reverse order addressing

Table Access to Prog Memory

- Allows a 64kB range of program memory to be accessible using special table instructions
- This is a modified Harvard architecture
- When using the table instructions:
  - The 24b program word address is: TBLPAG<7:0>:EA<15:1>:0
  - To allow byte access, the even/odd part of the program word is selected by EA<0>

Table Access Examples

- Consider
  - MOV #0, W0
  - MOV W0, TBLPAG
  - MOV #0x0294, W1
  - TBLRDL [W1], W0
- W1 = 0b**0000 0010 1001 010**0
- 24b program word address is 0xb0000 0000 **0000 0010 1001 010**0 = 0x00294
- LSW at this address is loaded into W0

Table Access Examples

- Consider
  - MOV #0, W0
  - MOV W0, TBLPAG
  - MOV #0x0294, W1
  - TBLRDH [W1], W0
- W1 = 0b**0000 0010 1001 010**0
- 24b program word address is 0xb0000 0000 **0000 0010 1001 010**0 = 0x00294
- MSW at this address is loaded into W0

Table Access Examples

- Consider
  - MOV #0, W0
  - MOV W0, TBLPAG
  - MOV #0x0295, W1
  - TBLRDL.B [W1], W0
- W1 = 0b**0000 0010 1001 010**1
- 24b program word address is 0xb0000 0000 **0000 0010 1001 010**0 = 0x00294
- MSB of LSW at this address is loaded into the LSB of W0

Program Flow Control (Jumps)

- Unconditional, Absolute
  - GOTO #*lit23*; RTL: PC ← *lit23*
  - Anywhere in memory, 2 word instruction
- Unconditional, Relative
  - BRA #*slit16*; RTL: PC += 2**slit16*
  - Range of +/- 32k instructs, 1 word instruction
- Programmatic, Indirect
  - GOTO Wn; RTL: PC = Wn

Program Flow Control (Jumps)

- Conditional, Relative
  - BRA *cond*, #*slit16*;
  - RTL: if (*cond)* PC += 2**slit16*
  - Range of +/- 32k instructs, 1 word instruction

Program Flow Control (Loops)

- Single-instruction repeats
  - REPEAT #lit14 or REPEAT Wn
  - Executes the next instruction 1+ *(#lit14* or Wn) times
  - Next instruction not flow control instruction
- Subroutine repeats
  - DO #lit14, Expr or DO Wn, Expr
  - Executes code up to PC+Expr 1+ *(#lit14* or Wn) times

Program Flow Control (Calls)

- Absolute
  - CALL #*lit23*
  - RTL: PUSH PC; PC ← *lit23*
  - Anywhere in memory, 2 word instruction
- Relative
  - RCALL #*slit16*;
  - RTL: PUSH PC; PC += 2**slit16*
  - Range of +/- 32k instructs, 1 word instruction
- Programmatic, Indirect
  - RCALL Wn; RTL: PUSH PC; PC = Wn

Interrupt Vectors

- Interrupt
  - 126 interrupt vectors (space for 8 non-maskable traps and 118 maskable interrupts)
  - 7 priority levels
  - Main IVT base address at word 4
  - Alternate IVT (AIVT) for debugging base address at word 104h
  - 24-bit address of interrupt service routine stored at address (base + 2 * n); n = interrupt #
  - The DISICNT register allows interrupt priority 1-6 to be disable for a specified number of counts

Interrupt Context Saving

- When an interrupt is invoked, PC, IPL3(CORCON<3>), and SRL are pushed on the stack and PC is loaded with the address in the IVT
- RETFIE restores this information
- PUSH.s automatically stores W0-3 and N, OV, Z, C, and DC in their shadow registers for fast context savings
- POP.s restores this information
- Only one level deep shadow register set

DMA Engine

- 8 prioritized channels of DMA
- Byte or word transfers
- Single or block transfer modes
- Allows transfer of data to and from peripherals such as serial ports and a/d converters without processor intervention or processor overhead in many cases