Heavy Metal Kata

Brief introduction to 8086 assembler

Main Registers

- **Accumulator** (A)
  Arithmetic
- **Counter** (C)
  Used in loops and shift rotate operations
- **Data** (D)
  I/O operations
- **Base** (B)
  Pointer to data

16 bit: AX, BX, CX, DX, 8 bit: parts L,H

Index Registers

- **Source Index** (SI)
  Used as a source pointer in string operations (in DS)
- **Destination Index** (DI)
  Used as a destination pointer in string operations (in ES)
- **Base Pointer** (BP)
  Points base of the stack
- **Stack Pointer** (SP)
  Points the top of the stack

All are 16 bit

Segment Registers

- **Data Segment** (DS)
- **Extra Segment** (ES)
- **Stack Segment** (SS)
- **Code Segment** (CS)

All are 16 bit

Program Counter

- **Instruction Pointer** (IP)

Status Register

- **Carry** (C) Set when carry (borrow) was generated in last arithmetic operation
- **Zero** (Z) Set when last arithmetic operation result was zero
- **Sign** (S) Set when most significant bit is 1 (number is negative in two's complement form)
- **Overflow** (O) Set when 127 + 127 = -2 (in two's complement form)
- **Parity** (P) parity of the result of last operation
- **Adjust** (A) - half carry (carry on lower 4 bits)
- **Interrupt** (I) - if 0 maskable interrupts will be ignored
- **Direction** (D) - 0 mean string is processing from lowest to highest address

Data transfer instructions

- **MOV** move data between:
  - Register: **MOV AX, BC**
  - Immediate: **MOV AX, 0x10**
  - Memory: **MOV AX, ES:[BX]** (MOV AX, ES:[BX+5],MOV AX, ES:[BX][SI]+5)
- **PUSH/POP** stack operations
  - Register: **PUSH AX**
  - Memory: **POP ES:[BX]**

Aritmetic

- **ADD/SUB** adding subtracting: (ADC/SBB - carry/borrow version)
  - **ADD** destination, source (destination = destination + source)
  - **ADD** ax, 10
  - **ADD** ax, bx
  - **ADD** ax, es:[bx]
  - **ADD** es:[bx], ax
- **INC/DEC** increasing/decreasing register or memory
- **MUL/DIV** multiply/divide source by accumulator
- **AND/OR/XOR/NOT/TEST** - binary operations
- **SHR/SHL/ROL/ROR** - shift/rotate

Control

- **CALL/RET** push/pop ip to stack
- **JMP** - Unconditional jump
- **J*** - paired with **CMP** or after arithmetic operations - conditional jump
- **LOOP** - decrease CX, and JNZ

Table 2-15. Interpretation of Conditional Transfers

MNEMONIC	CONDITION TESTED	"JUMP IF ..."
JA/JNBE	(CF OR ZF)=0	above/not below nor equal
JAE/JNB	CF=0	above or equal/not below
JB/JNAE	CF=1	below/not above nor equal
JBE/JNA	(CF OR ZF)=1	below or equal/not above
JC	CF=1	carry
JE/JZ	ZF=1	equal/zero
JG/JNLE	((SF XOR OF) OR ZF)=0	greater/not less nor equal
JGE/JNL	(SF XOR OF)=0	greater or equal/not less
JL/JNGE	(SF XOR OF)=1	less/not greater nor equal
JLE/JNG	((SF XOR OF) OR ZF)=1	less or equal/not greater
JNC	CF=0	not carry
JNE/JNZ	ZF=0	not equal/not zero
JNO	OF=0	not overflow
JNP/JPO	PF=0	not parity/parity odd
JNS	SF=0	not sign
JO	OF=1	overflow
JP/JPE	PF=1	parity/parity equal
JS	SF=1	sign

The INT Instruction

Activate the interrupt procedure (I/O, BIOS, OS)

- INT 21h - dos interrupt
- INT 10h - bios interrupt

http://stanislavs.org/helppc/int_21.html

```
MOV AH, 0x02
MOV DL ‘x’
INT 0x21

MOV AX, 0x4c00
INT 0x21
```