# Exceptions Report

DifficultWare, CSS 422 A

This document contains the problems our team encountered but couldn’t fix. For each problem a description in detail of what we did to complete it and how we fell short in the allotted time is included.

## CMP, SUB, EOR With Immediate Value Disassembles as CMPI, SUBI, EOR

When EASy68K encounters the OP code CMP/SUB/EOR with an immediate value as the source operand, it optimizes the OP code and converts it as CMPI/SUBI/EORI when translating to hex values. Since CMPI/SUBI/EORI was not part of the OP codes to be disassembled, we assume that an immediate value will not be given as a source operand. The disassembler will treat the OP code as DATA and print the address of DATA.

## Problem Handling Unsupported Instructions With Additional Memory

If an unsupported instruction is passed with extra memory, there is a possibility that it will offset the pointer and mess up the rest of the disassembly. In our implementation, we assume that the unsupported OP codes are not followed by extra memory.

## Only Disassembles Valid 68k Opcodes

We define valid OP codes as the ones defined in the ‘Supported Instructions & Addressing Modes’ page on Canvas as well as the OP codes that EASy68k supports. If the OP code is not valid, then our disassembler will print a DATA line.

## MOVE Print Word When Immediate is a Byte

When MOVE is given an immediate value as a source operand, the disassembler prints four digits for both Byte and Word. For example, MOVE.B #$2, D0 is disassembled and displayed as MOVE.B #$0002, D0. We decided that this behavior was reasonable because the values are correct even with the leading 0s.

## MOVEM Prints All Registers Instead of Range

When MOVEM is given a list of registers in the form of AX-AY or DX-DY, it disassembles it as AX/A.../AY or DX/D.../DY, listing every register. We decided this behavior was reasonable because it would be difficult to state every range possible and it would be more accurate to list every register required.