Team Powerpuff report

CSS 422 - Vu Dinh and Alejandro Alfaro

Date: Nov 14 2014

To: CSS 422, Fall 2014

From: Vu Dinh and Alejandro Alfaro

Subject: Weekly team report #3 (opcode and <ea> decoding)

Work Completed:

- We've been putting our plans into action throughout the week. To elaborate:
  Our plan for decoding <ea> is to compare the given assembled value (e.g. 3C26) with known ranges of values for each <ea> mode. For example:

  MOVE instructions are known to be inside the range of $1000 to $3FFF (theoretically). The given value (3C26) can then be compared against known ranges for the **size** unknown. If it is between $1000 and $2000, then it is MOVE.B, so on and so forth.
  Once a piece of <ea> information has been determined, we truncate it from the source by subtracting away the value that it was discovered to be. For example, Once we know that 3C26 is a MOVE.W instruction, we subtract the WORD-size information value away ($3000). This gives us a value of $C26 to continue decoding for other fields. For instance, C indicates that the destination register is 6, and $26 being smaller than $100 indicates that the destination mode has to be data register. $26 also tells us that the source mode is indirect addressing, and the source register is 6.
  Thus, the whole instruction is decoded as: MOVE.W (A6), D6

  To do this, we need those "known ranges" and "special values" for each opcode. This is what we've spent the majority of our time on. We've finished it for over half the opcodes.

- Initially, we've planned to use the same approach for decoding opcodes, but the abundance of overlapping ranges is posing a serious trouble. To remedy this, we are coming up with a different plan to decode opcodes. This work is on-going. However, we have managed to come up with one idea: to chop the bits up and compare them to the opcode's signature values.
- Opcode/EA table as required (see bottom).

Problems encountered:

- A lot of opcode instructions have overlapping value ranges. While they are still unique, identifying opcode via value ranges is proving to be too troublesome. We are looking into easier alternatives. This problem does not affect our strategy for decoding <ea>.

Work scheduled for next week:

- Finish generating known values for all opcodes.
- Write Assembly code to decode <ea> for all opcodes.
- Conduct controlled testing (perhaps foregoing opcode decoding and go straight to testing <ea> decoding).
- Quickly coming up with ideas for decoding opcodes and evaluating them within the week. If all fails, the comparing-value-ranges method is still viable, however arduous.

Self-evaluation: We've gotten past the thinking stage and we are now physically busy typing up code, so I think we're still on schedule, especially considering how the Assembly programming "best practices" are still being introduced in class. The opcode decoding process is not going too well, which is disappointing, so I hope we'll get it resolved next week.

Opcode/EA table

We do not fully comprehend the request, but here's what we think you want.

Addressing mode	Syntax	Mode values	Register values	HEX range
Data Register Direct	D0-D7	000	000-111	00 to 07

Address Register Direct	A0-A7	001	000-111	08 to 0F
Address Register Indirect	(A0) - (A7)	010	000-111	10 to 17
Immediate Data	#<value>	111	100	3A
Address Register Indirect with post-increment	(A0)+ to (A7)+	011	000-111	18 to 1F
Address Register Indirect with pre-decrement	-(A0) to -(A7)	100	000 - 111	20 to 27
Abs long address	(xxx).L	111	001	39
Abs word address	(xxx).W	111	000	38