Supplementary Material

1 Granularity of Data Inlining and Instruction Set Interleaving

The binaries in SPEC-Inter contain coarse-grained ARM/Thumb interleaving. In our evaluation, we do not introduce more fine-grained ARM/Thumb interleavings (e.g., within functions) considering the following reasons.

First, in stripped binaries, function entries are unknown such that our disassembler and the baselines have to deal with the code blob as a whole.

Second, instruction mode changes happening at function calls are similar to those within functions and do not degrade the difficulty for disassembly. As described in Section 2.2 in the paper, the instruction mode can be changed by some branch instructions, e.g., blx. Unlike the call instruction in x86/x64, there is no specific function call instruction in ARM binariesA function call in ARM is also considered as a branch, using the `bl`/`blx` instruction. Thus, the mode changes at different positions are similar. Note that the ARM/Thumb interleaving shown in the motivation example (Figure 1 in the paper) is a tail call instead of being in the middle of a function.

In addition, we build SPEC-Inter by compiling different files/modules with different instruction sets, which is a common process when building real ARM applications. Crafting binaries with a lot of ARM/Thumb interleavings within functions can be achieved at the assembly code level, but hard at the compilation level.

As for SPEC-Data, we do not embed data into the middle of functions. However, we do this for the obfuscation dataset.

2 Failure Rate of P-Disasm and Spedi

As discussed in Section 3.2 in the paper, P-Disasm is based on an assumption that the occluded instruction sequences tend to quickly converge on true instructions. However, this is only true for x86/x64 binaries. For ARM binaries, as there are two instruction sets, the occlusion of instruction sets exists in the entire code sections. Without the help of the converged instructions, there is no local propagation within the occlusion space, and the propagation algorithm becomes quite slow and consumes a huge amount of memory. As shown in Table 1, P-Disasm terminates for more than half of binaries on all datasets due to memory explosion.

Spedi also has issues with disassembling large binaries, especially the complex obfuscated binaries. Note that Spedi only supports Thumb instructions and are only evaluated on binaries compiled with Thumb instructions, i.e., the half of SPEC-Basic, SPEC-Data, and the binaries built with obfuscation.

3 More Details of Efficiency

XDA. Although XDA is quick in disassembly, it requires additional time for model training.In our evaluation, the XDA model trained on SPEC CPU2006 takes 21 hours for pretraining and 15.6 hours for finetuning, following the default settings of 10 epochs and 30 epochs, respectively.

D-ARM. Figure 1 plots the relation between the time cost of D-ARM and the binary sizes (on the AOSP dataset). The relation tends to be linear, which implies that D-ARM can scale gracefully in practice.

Table 1: Failure Rate (%) for P-Disasm and Spedi

Dataset		P-Disasm	Spedi
Basic	SPEC2000	54.58	5.42
	SPEC2006	57.29	13.33
Data	SPEC2000	51.67	9.58
	SPEC2006	58.54	12.92
Inter	SPEC2000	60.63	N/A
	SPEC2006	65.83	N/A
$r = 0$	SPEC2000	57.08	77.50
	SPEC2006	60.67	72.27
$r = 50$	SPEC2000	64.58	84.17
	SPEC2006	63.60	81.51
$r = 100$	SPEC2000	75.83	89.17
	SPEC2006	67.36	81.51

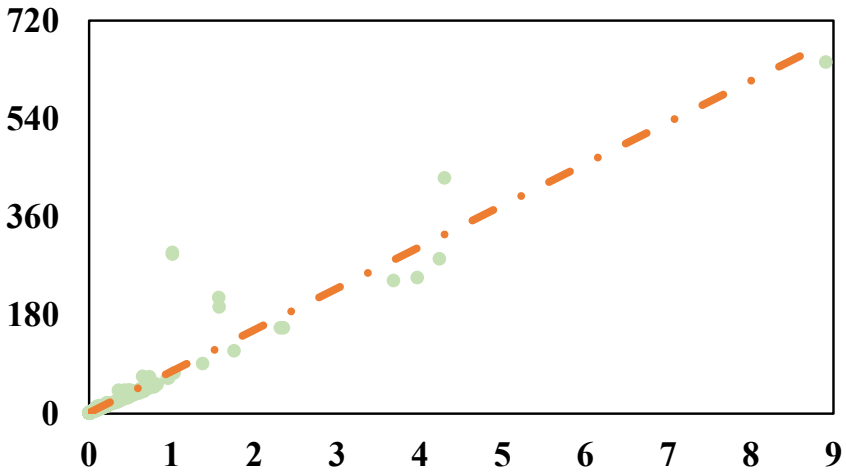

Figure 1: Time cost (Y-axis, seconds) taken by D-ARM exhibits a linear complexity in binary size (X-axis, MB).