UNITED STATES DISTRICT COURT
WESTERN DISTRICT OF TEXAS
WACO DIVISION

INTERNATIONAL BUSINESS MACHINES CORPORATION, Plaintiff, v. LZLABS GMBH, and TEXAS WORMHOLE, LLC, Defendants.	Civil Action No.: **DEMAND FOR JURY TRIAL**

COMPLAINT

Plaintiff International Business Machines Corporation ("IBM") files this Complaint against Defendants LzLabs GmbH ("LzLabs") and Texas Wormhole, LLC ("Texas Wormhole") (collectively, "Defendants") and alleges as follows, based on knowledge as to itself and its own acts and on information and belief as to all other matters except as indicated otherwise.

INTRODUCTION

1. Once again, IBM must bring a lawsuit against a company owned and controlled by John Moores ("Moores") to stop it from misusing IBM's intellectual property. LzLabs is a company based in Switzerland, controlled by Moores, and run by Moore's long-time business associate, Thilo Rockmann ("Rockmann"). LzLabs was formed by Moores in 2011, shortly after Moores and his prior company were enjoined by this Court for engaging in a scheme to free-ride on IBM's mainframe business. Now, LzLabs is the vehicle through which Moores and Rockmann are attempting to engage in another free-riding effort.

2. Moores and Rockmann's first order of business for LzLabs was to figure out how it could gain access to IBM mainframe software. To acquire that access, LzLabs set up a shell entity to license the IBM mainframe software from a subsidiary of IBM (IBM UK). This

shell entity is called Winsopia. Winsopia has no business, except to act at the direction of LzLabs. And that direction is to engage in improper reverse engineering of the IBM software to gain IBM's trade secret and proprietary information. LzLabs then uses this information to develop a product offering that LzLabs claims is a plug-and-play replacement for the very IBM offerings LzLabs deceitfully obtained – IBM's industry-leading mainframe system software.

3. LzLabs' alleged plug-and-play replacement for IBM's mainframe system software is called the Software Defined Mainframe ("SDM"). LzLabs claims its SDM can run customer owned software applications written for IBM mainframes and process the related data without making modifications to the code or data for those applications, thereby, (according to LzLabs), duplicating the functionality of IBM mainframe systems.[1] While IBM has committed decades of engineering effort and billions of dollars of investment to develop its industry-leading mainframe systems, LzLabs claims to have achieved this feat in a fraction of the time and with a fraction of the engineers IBM used.

4. After IBM UK learned of the connection between the shell entity Winsopia and LzLabs (and to ensure compliance with the agreements by which the shell entity licensed the mainframe software), IBM UK exercised its audit rights under its agreements with Winsopia. Winsopia, however, refused to comply with the audit request, even though IBM UK's audit rights under the agreements are not discretionary and do not require assent. In fact, before it would even entertain the audit request, Winsopia demanded that IBM UK sign a non-disclosure agreement that effectively required IBM to waive its legal rights before Winsopia provided any of the requested information.

[1] Even if the SDM could run IBM mainframe applications without modification, which it likely cannot, the SDM cannot provide the same reliability, security, availability and performance as the IBM mainframe systems.

5. Despite the efforts LzLabs has taken to conceal its wrongful activity, IBM has recently collected what information it can find, and as detailed below, has determined what logic dictates, *i.e.*, that LzLabs could not have developed its SDM offering with the specific functionality it claims without misappropriating IBM's intellectual property.

6. Remarkably, as mentioned above, this is not the first time Moores and Rockmann have attempted to free-ride on IBM's mainframe business and cover it up. The first attempt by the pair to misappropriate IBM's intellectual property was through a company called Neon Enterprise Software, LLC ("Neon"). Moores controlled Neon, and Rockmann ran Neon's European sales efforts.

7. In 2009, Neon introduced a product named "zPrime." zPrime modified the operation of IBM's mainframe systems to enable customers to offload processing from certain IBM mainframe processors and, in turn, to reduce the fees customers owed to IBM for IBM software. The modifications were a direct violation of the customers' contracts with IBM, and Neon knew it. In fact, at the time, Rockmann admitted that zPrime went "against what IBM intended their systems to do."[2] Nonetheless, Neon proceeded with its efforts, believing it could generate a revenue stream based on a percentage of the fees its software allowed customers to save – even though the generation of that revenue stream was illegal, because it depended on inducing IBM's customers to breach their agreements with IBM by misappropriating computing capacity.

8. Neon's scheme also depended on its misuse of IBM's intellectual property. Unable to develop the zPrime software legally, Neon, having acquired an IBM mainframe

[2] Peter Judge, *IBM Tries To Stall Neon's zPrime Mainframe Booster*, Silicon.co.uk (Aug. 28, 2009), https://www.silicon.co.uk/e-enterprise/financial-market/ibm-tries-to-stall-neons-zprime-mainframe-booster-1708 (last visited February 17, 2022).

system (including the operating system software), illegally reverse-engineered IBM software to discover and use IBM proprietary information, made illegal copies of IBM software, and caused users of its zPrime software to do so as well – all in violation of IBM's intellectual property rights and the agreements under which Neon licensed the IBM mainframe software.

9. In 2010, IBM brought claims against Neon for tortious interference with IBM's contracts with its customers, breach of Neon's own license to IBM's mainframe software, copyright infringement, and Lanham Act violations. *See Neon Enterprise Software, LLC v. International Business Machines Corp.*, Case No. 1:09-cv-00896 (W.D. Tex.). Then in 2011, this Court entered a permanent injunction against Neon, Moores, and a number of Neon executives. *Neon Enterprise Software, LLC v. International Business Machines Corp.*, Case No. 1:09-cv-00896 (W.D. Tex.), Dkt. 165 (May 31, 2011). The permanent injunction resulted not only from Neon's misuse of IBM's intellectual property rights and breach of Neon's contractual obligations to IBM, but also from Neon's blatant attempt to hide its wrongdoing by destroying evidence and misrepresenting its conduct—under oath—even after the litigation was filed.

10. Not to be deterred, even after this Court stopped them once, Moores and Rockmann have picked up where they left off, with LzLabs as the "new Neon." While LzLab's endgame is the same as Neon's—to free ride on IBM's mainframe business—it is doing so in a different way. Rather than divert a portion of the revenue owed by customers for their use of IBM mainframe computing resources, LzLabs engaged in a scheme to replace the entire IBM mainframe system. LzLabs could have competed legally for these computing workloads, but it did not even attempt to do that. But like Neon before it, LzLabs chose a path that violated IBM's rights. LzLabs' efforts should end the same way Neon's efforts ended – with this Court ordering

LzLabs to stop its wrongful conduct and properly compensate IBM for the misuse of its intellectual property.

JURISDICTION AND VENUE

11. The Court has jurisdiction over this action under 28 U.S.C. §§ 1331, 1338, and 1367.

12. Personal jurisdiction is proper against LzLabs because it has engaged in substantial activities in the United States in connection with the development, marketing, and offers for sale of its SDM. LzLabs engages in widespread marketing efforts of its SDM in the United States.[3] LzLabs also advertises several major partnerships providing access to its SDM through the cloud.[4] Moreover, LzLabs has made direct offers for sale and engaged in numerous discussions concerning the operation of the SDM with at least one IBM mainframe customer in the United States.

13. LzLabs has engaged in development work for its SDM in the United States, including within Texas. Defendant Texas Wormhole is LzLabs' Austin, Texas-based development arm, engaging in development of the SDM in this District. Texas Wormhole acts under the direction (and for the exclusive benefit) of LzLabs. By way of example, Texas Wormhole employees Steve Towns[5], Gary Trinklein[6], Tom Harper[7], and Tommy Sprinkle[8]

---

[3] *See*, *e.g.*, https://www.lzlabs.com/resources/lzlabs-expands-into-north-american-market-liberating-legacy-applications/ (last visited February 17, 2022).

[4] *See*, *e.g.*, https://www.lzlabs.com/resources/lzlabs-expands-into-north-american-market-liberating-legacy-applications/ (last visited February 17, 2022).

[5] (https://www.linkedin.com/in/steve-towns-95908218/) (last visited February 17, 2022)

[6] (https://www.linkedin.com/in/gary-trinklein-b5480942/) (last visited February 17, 2022)

[7] (https://www.linkedin.com/in/tom-harper-57537216/) (last visited February 17, 2022)

[8] (https://www.linkedin.com/in/tommy-sprinkle-50a63599/) (last visited February 17, 2022)

reside, according to their respective LinkedIn profiles, in Texas, and all have roles relating to software development, such as software engineer, developer, or architect. Tom Harper is expressly named in the Neon injunction. These individuals' work for LzLabs includes development of the LzLabs SDM including the reverse assembling, reverse compiling, translating, or reverse engineering of IBM software to develop the SDM.

14. Venue is proper in this District under 28 U.S.C. § 1391 and 1400. For example, Texas Wormhole has a regular and established place of business in this District and engages in development and use of the SDM in this District, actions that infringe the patents asserted in this action. LzLabs has directed Texas Wormhole's misappropriation in this District. Furthermore, because LzLabs is a foreign corporation subject to personal jurisdiction in the United States, venue is proper in any United States District Court for the causes of action asserted here.

THE PARTIES

15. Plaintiff IBM is a corporation organized and existing under the laws of New York, with its principal place of business in Armonk, NY. IBM designs, manufactures, sells, and licenses computer hardware and software, and provides related services.

16. Defendant LzLabs is a company organized and existing under the laws of Switzerland with its principal place of business at Richtiarkade 16, 8304 Wallisellen, Switzerland. LzLabs purports to be a software development company.

17. Defendant Texas Wormhole, LLC is an LLC organized and existing under the laws of Delaware, with its principal place of business at 111 Congress Avenue, Suite 2600, Austin, TX 78701. It is an affiliate of LzLabs and works on the development of LzLabs' SDM at the direction of, and for the benefit of, LzLabs.

FACTS

IBM Mainframes

18. As part of its server and hybrid cloud offerings, IBM designs, manufactures, and sells IBM mainframe systems that are extremely reliable and secure and that experience minimal downtime and that are used by IBM's customers, including large companies, governments, and organizations for a variety of critical computing work. These systems include hardware, such as the mainframe servers and storage devices, as well as operating systems and other software designed, developed and licensed by IBM. IBM mainframe systems can store massive amounts of data and process billions of calculations and transactions in real time. They are critical to commercial databases, transaction servers, and other applications that require high resiliency, security, and agility. IBM customers use their mainframe systems for a wide range of tasks such as processing customer orders, executing secure and voluminous financial transactions, managing payrolls, and tracking inventory. IBM's current line of mainframe computers consists of its IBM Z models, the most recent being the IBM z15.

19. IBM's mainframe computer servers face intense competition from many on-premises and cloud server offerings, including Linux and Microsoft Windows-based systems executing on x86-based computer architectures, with respect both to retaining existing computing workloads and attracting new workloads. IBM mainframe systems represent a small portion of all servers, whether on-premises or cloud based. IBM has continued to invest in its Z mainframe computer servers and software to provide customers a computing platform with the highest security, performance, and availability. For example, IBM's continued investment has resulted in important innovations in areas such as cybersecurity, encryption and artificial intelligence.

20. IBM welcomes competition, believing that competition drives innovation and results in improved performance, value, and reliability for its customers. However, that competition must be lawful. IBM brings this suit to address LzLabs' unlawful development of its SDM and illegal use of IBM's software.

The Proprietary IBM Multi-Level Software Environment

21. As described more fully below, IBM has created a multi-level software environment, based on a proprietary mainframe Instruction Set Architecture, to operate on its mainframe computers. IBM developed the collection of software that forms this multi-level software environment through extensive software engineering and research and development efforts over a period of approximately 60 years and an investment of billions of dollars. The extensive effort and investment have resulted in a secure and proprietary multi-level software environment that is protected by valuable intellectual property rights of IBM.

22. Figure 1, below, shows the levels of the IBM mainframe software and hardware environment. The top four levels represent software, and the fifth and lowest level represents the mainframe computer hardware (which is a combination of hardware and specialized software). The top level of the diagram shows customer applications, which are programs that are generally developed for the specific needs and business of a customer. The next level down shows the compilers and run time services that are used to turn the language used to write the customer application (often referred to as source code) into code that can be run on the hardware itself, together with run time services. The third level down shows the middleware services that provide necessary functions (such as database and customer transaction processing functions) necessary to support the customer application. The fourth level down shows the operating system layer, which provides additional services and access to the

underlying hardware functions. And the bottom level shows the processing of the instructions that are run on the processors of the hardware itself.

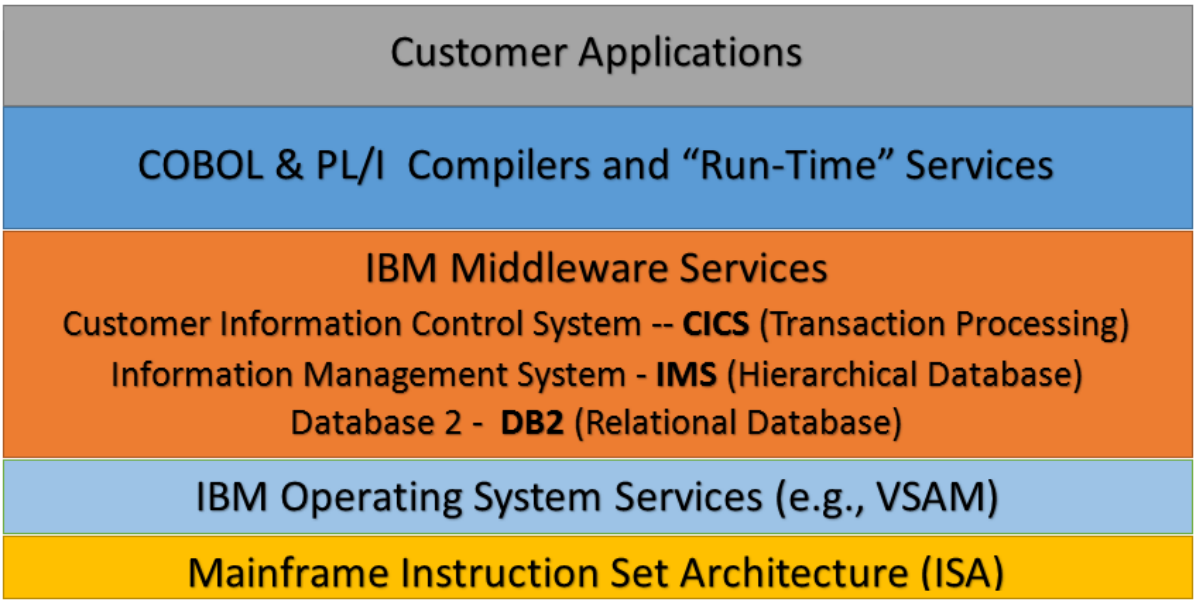

Fig. 1: IBM Mainframe Environment

23. The software programs or "applications" shown in the top layer can be written by IBM, independent software vendors, or IBM customers to perform user-oriented tasks. Application software programs perform specific functions for users, such as processing financial transactions or payroll management. These are typically programs with which the customer interacts and that provide functionality developed or tailored specifically for a complex business need of the customer – such as secure banking transactions, shipping logistics, or a worldwide airlines reservation system. Most mainframe applications need to be constantly on-line (meaning very little downtime), secure, fast, and capable of processing large amounts of data.

24. Customer applications for use on the IBM mainframe (such as those described above) are typically written in high-level (somewhat English-like) programming languages, such as COBOL (**CO**mmon **B**usiness **O**riented **L**anguage) or PL/I (Programming Language One). A

program as written by a programmer (in COBOL, PL/I, or another programming language) is generally referred to as the "source code" for the program.

25. The next level, *i.e.,* the second layer in Figure 1, includes "compilers" and "run-time services." Programs called "compilers" translate source code that is written in a language such as COBOL or PL/I into "machine language" instructions that the computer ultimately processes. Like any computer, an IBM mainframe computer contains various hardware components, including processors (which perform computations and execute instructions) and memory (which stores data used by the computer). These components only understand data and instructions in binary form (*i.e.*, in "1"s and "0"s). Moreover, the processors only understand and process a particular set of instructions, and accordingly, every program that a computer processor executes must be comprised of instructions in the language that computer processor is programmed to run. These instructions are commonly referred to as "machine language." The machine language produced by a compiler (such as the COBOL compiler shown in Figure 1) is typically referred to as the "object code" for the program. After compilation, the resulting object code is organized[9] into what is called an executable load module that interacts with the compiler run-time services and with the middleware layer, the operating system layer, and the processors that execute the machine instructions, each of which is described below. The executable load module may be repeatedly executed without re-compiling the source code.

26. Compilers generally translate source code instructions written by the programmer directly into sets of machine language instructions. However, for certain source code instructions—typically those that require complex sets of machine language instructions—

[9] This process involves binding (or "link-editing") the object code for customer-written programs together with the object code for various IBM-written licensed programs.

compilers embed in the object code sets of machine language instructions that result in "calls" to certain pre-coded auxiliary service routines. IBM's COBOL and PL/I compilers use this technique, and such service routines are generally referred to as IBM's "COBOL & PL/I Runtime Service Routines." Each COBOL & PL/I Runtime Service Routine is a program or set of programs that includes valuable code created by IBM software engineers. COBOL & PL/I Runtime Service Routines are included in a software library that is provided to customers under a license that restricts what can be done with them. Unlike the Middleware Service Routines described below, the requests for which are generally coded by the customer application programmer, calls to COBOL & PL/I Runtime Service routines are typically inserted by the compilers and operate within the customer application program without the need for the customer application programmer to be aware of or actively involved with the process of embedding the requests for, or "calls" to, these service routines. Determining all the non-public detailed information regarding the COBOL & PL/I Runtime Service Routine code can only be accomplished through substantial effort that is prohibited by the license agreement covering these services.

27. Middleware products form the next level down, *i.e.*, the third level from the top, in Figure 1. The listed IBM middleware products are specialized programs, the primary purpose of which is to provide commonly used services to customer application programs, as well as to assist in setting up the complex environments necessary for the operation of customer applications. As indicated in the diagram, the IBM middleware programs that contain such "Middleware Service Routines" that IBM has developed for use with its IBM Z mainframe machines include IBM's Customer Information Control System ("CICS"), its Information Management System ("IMS"), and its Db2 products, each of which is described below.

28. CICS is a middleware service that provides a general-purpose transaction-processing subsystem that can be utilized by the customer applications to provide necessary functionality to run an application online. For example, applications can be made accessible to users from local or remote workstations, so that the application can process requests from one user while other users of that application can submit requests that utilize the same programs, files, and data at the same time the original request is being processed.

29. IMS provides database services commonly used to process online transactions, as well as transaction processing services like those provided by CICS. The database services provided by IMS involve databases in which the data is organized hierarchically. Those provided by Db2 involve "relational" databases, in which the data is organized in sets of related tables.

30. These middleware products provide customer applications with the ability to call a large variety of IBM software modules to provide functionality commonly needed to implement the types of customer applications generally run by IBM mainframes, without the need for the customer application developer to write all the complex code necessary to implement such functionality.

31. Like COBOL and PL/I Runtime Service Routines, each IBM Middleware Service Routine is a valuable program or set of programs that represents the fruits of extensive coding and research and development efforts by IBM software engineers over the course of many years. Each such Middleware Service Routine serves as a software module that can be invoked by a customer application, which—among other benefits—saves customer application developers from having to write the code that is contained in the module. When the customer application developer writes a customer application, she can include language that will, after

translation and compilation, result in binary code that sets up the input for, calls, and receives the output from a given Middleware Service Routine providing the commonly desired functionality.

32. IBM provides its Middleware Service Routines to customers for a fee and pursuant to a license that restricts what can be done with them. Customers who pay for a license to middleware containing such Middleware Service Routines are provided with high-level descriptions of the operation of each Middleware Service Routine and an Application Programming Interface ("API") specification of how to write, in a customer application program, source code that will result in binary code that invokes the desired service provided by the middleware, but IBM ordinarily does not provide them with a detailed description of the Middleware Service Routine code or architecture, or with all the detailed information about how the Middleware Service Routine code interacts with compiled application code. Determining all the non-public, detailed information regarding the Middleware Service Routine code can only be accomplished through substantial effort that is prohibited by the license agreement covering these services.

33. At the next level down in Figure 1, *i.e.*, the fourth level from the top, the operations of the IBM mainframe environment are coordinated by one or more operating systems ("OS"). OS programs provide additional callable services and manage a computer system's internal workings (often referred to as "hardware"), including the allocation, use, and management of memory, processors, connection paths for input and output, devices, and file systems. IBM's primary proprietary operating system for its mainframe computers is called "z/OS," which IBM first released in 2001. It is compatible with a range of commercial and open-source application software programs and is designed to be backward compatible with older systems and software. OS programs, such as z/OS, also provide an additional set of

services (called Operating System Services) that can be used by customer application programs. Examples of such services are those provided by the VSAM (Virtual Storage Access Method) component of z/OS, which includes services to assist customer application programmers in performing complex tasks such as reading and writing information to various external data storage devices, such as tape drives, disk drives, and other, more complex mass storage facilities that house the enormous amounts of data that are processed by today's mainframe computer systems. As with the middleware services, customers who pay for a license to IBM's z/OS operating system containing such Operating System Services are provided with high-level descriptions of the operation of each available Operating System Service and an API specification explaining how to set up the input for, call, and receive the output from each such service in a customer application program. But IBM ordinarily does not provide customers with a detailed description of the Operating System Service's code or architecture or all the ways it interacts at a binary level with compiled application code. Once again, determining all the non-public detailed information regarding the Operating System Service code can only be accomplished through substantial effort that is prohibited by the license agreement covering these Operating System Services.

34. The bottom layer of the IBM mainframe environment, the lowest level in Figure 1, is the hardware itself.[10] The IBM mainframe hardware is designed to provide a complex and specific Instruction Set Architecture ("ISA") executed on the processors, and like other competing hardware platforms, it will only execute programs composed of machine instructions that conform to that architecture. IBM's proprietary mainframe ISA has been

[10] The hardware includes embedded software referred to as Licensed Internal Code, millicode, or microcode. Together, this hardware and embedded software comprise the machine interface as seen by the upper software layers.

steadily extended and improved over approximately 60 years to the point where the ISA now includes about 1200 instructions, along with a complex, and similarly evolved, architecture infrastructure that is necessary for these 1200 instructions to properly operate.

35. As noted above, the IBM Compilers (along with their associated run time services), the IBM Middleware Service Routines, and the IBM Operating System Services are provided to a customer only pursuant to a license agreement. Such license agreements allow the customer to utilize the software only in certain prescribed ways and for certain purposes. These license agreements also prohibit certain activity with respect to the licensed software. Among other things, the license agreements prohibit the reverse assembly, reverse compilation, translation and/or reverse engineering of the IBM software. This in turn protects the IBM trade secrets embodied within such software that have resulted from decades of development and the investment of billions of dollars. Generally, the IBM software described above is provided to the customer only in object (or binary) form. It is a string of 1s and 0s that are not, in any meaningful sense, readable by humans. However, reverse assembly, reverse compilation, translation, and/or reverse engineering of the IBM binary software could enable someone to discern the proprietary structure, form, operation, and implementation details of the IBM software, and use of this information would amount to an ill-gotten benefit of the investments IBM has made over many years to develop such software. Because this is a common issue with software, software today is often provided to users only in object form and under license agreements containing similar prohibitions on reverse assembly, reverse compilation, translation and/or reverse engineering.

36. Many of the various internal components of the IBM software and hardware architecture are trade secrets, and IBM's license provisions that prohibit the reverse assembling,

reverse compiling, translating, or reverse engineering of IBM object code are a crucial element of their protection. Although certain information about external aspects of the software and related architecture may be provided to customers, core information about internal aspects of the software is typically not made available to customers and is not otherwise generally known to, nor ascertainable by authorized means by customers or other persons who can obtain economic value from their disclosure or use.

37. IBM holds numerous patents resulting from the development of its mainframe system. Many of these patents are related to its mainframe ISA, operating systems, middleware, and software applications, as well as patents relating specifically to computer programs for emulation and translation procedures that provide efficiency in the complex mainframe environment or non-mainframe environments where mainframe hardware or software emulation is performed.

38. As explained below, in the development, sale, offer for sale, importation, installation, and operation of its "Software Defined Mainframe," LzLabs misappropriated valuable IBM trade secrets and has infringed the five IBM patents discussed further below.

LzLabs' Software Defined Mainframe

39. Defendant LzLabs claims it has developed what it calls a "Software Defined Mainframe" or "SDM." LzLabs describes the SDM as a "platform" that includes, at least, components LzLabs refers to as "LzBatch," "LzOnline," "LzRelational," "LzHierarchical," "LzWorkbench," and "LzVault."[11] LzLabs also offers SDM-related services including "LzDiscover" and "LzEnable."[12] Information concerning LzLab's SDM offering in this Complaint comes largely from LzLabs' own marketing of its SDM.

40. In contrast to the IBM mainframe software that operates on IBM's mainframe hardware, the SDM operates on an x86 architecture platform, with processors sold by Intel or AMD, and running a Linux operating system. LzLabs claims that the SDM provides functionality that is equivalent to that of the IBM mainframe and software environment, but on the x86 and Linux platform.[13] Specifically, LzLabs claims that customer applications written and compiled for the IBM mainframe environment, such as that described above, can be operated on the SDM "without changes and without compromise to performance."[14] [15]

41. A key part of LzLabs' "sales pitch" for its SDM appears to be a claim that "modernization" of "legacy" mainframe applications is very difficult or impractical on the

---

[11] lzlabs.com/resources/lzlabs-gotthard-opens-escape-route-for-trapped-mainframe-users/ (last visited February 17, 2022)

[12] https://www.lzlabs.com/resources/how-to-produce-a-mainframe-migration-plan-lzenable/ (last visited February 17, 2022)

[13] *What IS a Software Defined Mainframe?* (October 8, 2017), https://www.linkedin.com/pulse/what-software-defined-mainframe-dale-vecchio/ (last visited February 17, 2022).

[14] *LzLabs Works with COBOL-IT to Shift Customer Mainframe Applications to Open Source Software*, LzLabs (Apr. 28, 2016), https://www.lzlabs.com/lzlabs-partners-with-cobol-it-to-enable-seamless-mainframe-application-migration/ (last visited February 17, 2022).

[15] *LzLabs Unveils World's First Software Defined Mainframe,* LzLabs (March 14, 2016), https://www.lzlabs.com/lzlabs-unveils-worlds-first-software-defined-mainframe/ (last visited February 17, 2022).

mainframe, and that LzLabs offers an "easier"[16] path to modernization by migrating mainframe applications to an x86-based Linux environment where it provides tools to assist the customer in modernizing parts of the application.[17] Contrary to this claim, however, IBM has made and continues to make significant investments in effective and efficient application modernization approaches that permit the customer to continue to take advantage of the mainframe's proven reliability, availability, serviceability, security, scalability, agility, and performance while also integrating with services running on other platforms to create what the industry terms a hybrid cloud environment.

42. If, however, the migration of an IBM mainframe application to another platform is desired, rewriting and recompilation of the application source code to adapt it for the new platform are required in almost all cases. Such migration may also require translation of underlying data used by the application to new data formats. LzLabs claims that the SDM can skip this process. The purported attraction of the SDM approach thus is the promise that the mainframe application can be migrated with "no recompilation" needed and without conversion of its mainframe data, which can be read and written in its "native formats." In other words, the customer application can run "as is" on the SDM platform, according to LzLabs.

43. In a May 2019 article, LzLabs' then CEO Mark Cresswell explained the LzLabs SDM as follows:

> The [mainframe] load module interacts with the operating system through the language environment, it never interacts directly. We've created a language environment that is compatible with the way the ones on the mainframe work, so the load module only ever talks to us—through this language lab—and then we

[16] *Mainframe modernization's knowledge transfer paradox*, LzLabs (December 16, 2021), https://www.lzlabs.com/resources/mainframe-modernizations-knowledge-transfer-paradox/ (last visited February 21, 2022)

[17] See, e.g., *A Graceful Path to Legacy Modernization*, an LzLabs White Paper, 2018 and LzLabs' *Mainframe Modernization Survey 2019*.

> simply turn around and use whatever underlying facilities are available to us—Postgress [*sic*], Linux, LDAP, and so on, to get the job done. That is how we deal with the fact that you've got all this mainframe stuff out there—we don't need to worry about it. We just have to present to the application the APIs that application might otherwise be using on the mainframe, in a language it understands.[18]

44. The primary targets of LzLabs' SDM are mainframe applications written in the COBOL and PL/I computing languages, including those that use IBM's middleware and operating system runtime services.[19] Unlike other migration services, which operate on the source code level, LzLabs claims to move applications seamlessly to Linux platforms—not in the COBOL or PL/I source code form that is written by the programmer (*i.e.*, the IBM customer), but rather at the compiled binary machine code level, which, together with certain IBM licensed code modules, makes up the load module or executable form of the program. In this way, LzLabs claims to be able to perform the migration without recompilation, even if a customer no longer possesses source code versions of its applications.

45. To accomplish this feat, *i.e.*, to duplicate the operation of the IBM mainframe and software environment in the way that LzLabs claims the SDM does, the SDM must mimic exactly the operation of core portions of the IBM software and architecture. The necessary mimicking includes, at a minimum, key portions of the IBM Operating System Service Routines, Middleware Service Routines, and COBOL & PL/I Runtime Service Routines outlined above. It

---

[18] Max Smolaks, *LzLabs kills Swisscom's mainframes – but it's not the work of a vicious BOFH: All the apps are now living on cloud nine*, The Register (May 16, 2019), https://www.theregister.com/2019/05/16/lzlabs_kills_swisscoms_mainframes/ (alteration in original) (last visited February 17, 2022).

[19] Although IBM believes that LzLabs' primary emphasis is support of COBOL and PL/I applications, LzLabs apparently claims some level of support for C language and Assembler language applications as well. This is reflected, for example, in the following LzLabs statement: "Interpretive execution is the basic mode of SDM operation. This approach enables the execution of legacy application programs written in COBOL, PL/I, C, and Assembler." [*The Anatomy of Mainframe Application Workload Migration*, an LzLabs White Paper, April 2018, p. 2.]

also includes emulation, or some other form of translation, of the application's IBM mainframe ISA instructions into x86 ISA instructions, the only instructions recognizable to the hardware processors on the x86 machines on which LzLabs claims its SDM runs.

46. To make this mimicry possible, LzLabs would need to have gained access to low-level information about the IBM Service Routines. Specifically, LzLabs would have required detailed information about the way the service routines interact with the compiled application code at a binary level and the required functional behavior of such routines. Due to the volume, complexity, and lack of publicly available information about certain aspects of such binary-level interaction and required functional behavior of the associated IBM service routines, LzLabs could not legitimately obtain such information without reverse assembling, reverse compiling, translating, or reverse engineering of IBM material. Such reverse assembling, reverse compiling, translating, or reverse engineering, at a minimum, would have required LzLabs or someone working on its behalf to undertake activities prohibited by the agreements under which IBM licenses its mainframe hardware and software. At a minimum, LzLabs or someone working on LzLabs' behalf engaged in improper reverse assembly, reverse compiling, or translating of the IBM software and Service Routines to convert object code versions of the IBM software into source code versions (or intermediate language versions) that can be read and understood by humans, and/or other types of improper reverse engineering of the IBM software. This would have allowed LzLabs software engineers to inspect – and copy – information that is kept as confidential trade secrets by IBM.

47. Examples of LzLabs' reverse assembling, reverse compiling, translating, and/or reverse engineering of the IBM Service Routines are discussed further below.

LzLabs' Improper Actions in Accessing, and Duplicating the Functionality of Key Portions of the IBM Software

48. As summarized above: (a) each IBM Operating System Service Routine and Middleware Service Routine includes APIs that define at the source code level what is required for an application program to (i) set up the input for; (ii) "call"; and (iii) receive the output from the Service Routines so that customer application programmers can reasonably write the source code to invoke a required service; (b) for all except a small number of the language-specific COBOL & PL/I Runtime Service Routines, such API information is not published because such services are not intended to be requested directly by the application programmer, but rather are invoked by object code inserted directly into the application program by, and at the discretion of, the IBM compiler; (c) the details of all the ways in which the service routines interact with the compiled application code at a binary level are generally not published, especially those for the language-specific COBOL & PL/I Runtime Service Routines; and (d) for all IBM Service Routines that are made available only in object code form, the details of the internal operation of the IBM Service Routine itself are simply not available.

49. LzLabs could not have achieved the level of compatibility it claims for the functionality it identifies for the SDM without having engaged in the types of prohibited reverse assembling, reverse compiling, translating, or reverse engineering described above. To achieve the level of compatibility LzLabs claims, it would have had to engage in this prohibited reverse engineering activity with respect to a large volume of the proprietary and secret details of the way in which the service routines interact with the compiled application code at a binary level and the internal operation of the IBM Service Routines. The various IBM software programs discussed above implicate hundreds of service routines, each with a different operational

characteristics and behavior. For example, across three generations of COBOL alone, there are hundreds of COBOL Runtime Service Routines. Moreover, the inputs, outputs, application binary interactions and operational characteristics of each of the service routines can have many variations, totaling thousands of combinations and permutations that LzLabs would need to have identified and duplicated to fully replace the IBM product service routines utilized by IBM mainframe applications and for which it claims to provide substitutes.

Exemplary Misappropriations of IBM Service Routines

50. Applications compiled and linked on an IBM mainframe using the IBM COBOL and PL/I toolchains contain or call, directly or indirectly, binary modules that are part of the IBM mainframe software environment. To support such applications on the SDM, the Defendants must have either executed the binary modules on the SDM or reverse assembled and/or reverse compiled and/or otherwise reverse engineered those binary modules to re-implement their functionality and the way in which they interact with the compiled application code at a binary level. All such activities are prohibited by the license agreements under which the IBM software is provided to customers.

51. Execution of an application developed and compiled on a mainframe using the IBM COBOL implementation involves complex, unexposed interactions between the compiled form of the application and the COBOL Runtime Services. Such interactions are not inherent in the COBOL language itself but reflect design choices that have been made over the years by the IBM teams that developed the COBOL Runtime Services and compiler. Such interactions are generally of little or no concern to an application developer lawfully developing COBOL applications on an IBM mainframe; however, to support existing customer COBOL applications

without the need for modification or recompilation of such applications, the developers of the SDM needed to understand and replicate the detail of these complex and unexpected interactions.

52. Gaining the necessary knowledge and understanding to replicate the detail of these complex and confidential interactions would have been impossible without reverse assembling, reverse compiling, translating, or reverse engineering parts of (a) the IBM binary modules link-edited into load modules for applications written in COBOL, (b) data blocks and code fragments inserted into application binaries by the COBOL compiler and toolchain, and (c) the COBOL Runtime Services, including code, data blocks, and their interactions.

53. By way of example, COBOL has a language feature known as 'declaratives,' one purpose of which is to allow application developers to specify application code that is to run in the event of an error or exception occurring during an attempt to perform a file input output (or "I/O") operation (*e.g.*, "read" or "write"). Because, for example, these error/exception declaratives are for handling unexpected I/O conditions they are also referred to as I/O declaratives. A single application may contain multiple I/O declaratives, each of which may be specific to a file or may apply to all files but be specific to a particular kind of I/O operation (*e.g.*, a "read").

54. The appropriate declarative-specified application code is called in the event of an error or exception (or other specified condition) by the COBOL runtime. The mechanism for resolving which declarative-specified code to call and when to do so is complicated and hidden from application developers because it is part of the implementation detail of the IBM COBOL compiler and runtime. The mechanism for resolving declaratives and the data structures used to implement them are not discernable except through impermissible reverse engineering because

they are the result of numerous design choices made by IBM developers over many years of development of the IBM software.

55. A second example of LzLabs' misappropriation concerns a widely used language feature of PL/I know as condition handling, which allows application developers to specify the condition handling code to which control is passed in the event of a specified condition (such as an error). Conditions can be qualified in numerous ways, and the same condition can be handled by different code depending on the state of the application when the condition is triggered. The number of permutations within an application is essentially unlimited. The PL/I runtime is responsible for determining which code to execute, as well as the parameters to use, upon the occurrence of a particular condition. The manner in which the PL/I runtime determines the handling code to execute and the parameters to use is hidden from application developers. The choices are undocumented and are the result of numerous design decisions made by IBM engineers over the course of decades. These design choices are discernable only in the human-readable version of the code for the IBM PL/I runtime routines, which, as discussed above, is not provided to the customer.

56. Therefore, the SDM would not be able to recreate this functionality without LzLabs' (or someone acting at LzLabs' direction) reverse assembling, reverse compiling, translating, or reverse engineering of IBM's implementation of this functionality within the PL/I runtime routines.

57. Numerous other examples of functionality within the IBM compilers, middleware, and service routines could only have been determined through impermissible reverse engineering. The IBM compilers, middleware, and service routines were developed by IBM over the course of decades and involved countless subjective design choices that cannot be

replicated based on publicly available information. The code necessary to implement such functionality is maintained as a trade secret by IBM, and LzLabs' improper reverse engineering of that code constitutes misappropriation of those valuable trade secrets.

IBM Patents Infringed by LzLabs

58. The extensive investment and development effort in the IBM mainframe software and hardware has not only resulted in valuable IBM trade secrets, but also a significant number of patents covering many innovations resulting from IBM's development of its mainframe systems. For example, IBM created novel solutions to address efficiency and performance problems relating to the emulation of IBM mainframe applications, including in connection with the development of its IBM Z Development and Test Environment, as well as novel additions to its mainframe Instruction Set Architecture. This action focuses on five such patents.

59. U.S. Patent No. 9,804,823 (the "'823 Patent"), entitled "Shift Significand of Decimal Floating Point Data," was issued by the United States Patent and Trademark Office on October 31, 2017. A true and correct copy of the '823 Patent is attached as Exhibit 1.

60. The '823 Patent provides systems and methods for efficiently using decimal floating point data and instructions within a processing environment. Floating point refers to an approach to representing numbers within a computing system in which the *radix point* (*e.g.*, decimal point in base 10 systems) can float, or move, with respect to the significant digits (*significand*) of the number, its location being controlled by a specified exponent applied to the numerical base, such as base 10 in decimal, of the floating point representation. The use of decimal floating point format to process decimal data has fewer limitations than other floating point formats. For example, it is not subject to the precision losses inherent in the use of binary

or hexadecimal floating point format to process such data. The ’823 Patent facilitates the use of decimal floating point instructions to perform calculations involving variables specified in other decimal formats, such as packed decimal. Specifically, the patent teaches efficient decimal floating point instruction formats and methods of operation for, *e.g.*, the operations of scaling up or down by powers of 10, adding or removing left or rightmost digits for alignment and/or truncation of decimal values, and multiplying or dividing by powers of 10. The ’823 Patent arose out of development efforts to improve the performance and efficiency of the IBM mainframe architecture. IBM is the current owner by assignment of all right, title, and interest of the ’823 Patent.

61. U.S. Patent 8,190,664 (the “’664 Patent), entitled “Employing a Mask Field of an Instruction to Encode a Sign of a Result of the Instruction,” was issued by the United States Patent and Trademark Office on May 29, 2012. A true and correct copy of the ’664 Patent is attached as Exhibit 2.

62. The ’664 Patent provides effective systems and methods for employing decimal floating point data and instructions within a processing environment. Among other things, the ’664 Patent provides efficient ways to compose/decompose data that is in a different decimal format into decimal floating point data format and, more specifically, an efficient way to manage the sign of the result of a decimal floating point operation when that result is converted into a different decimal format. IBM is the current owner by assignment of all right, title, and interest of the ’664 Patent.

63. U.S. Patent No. 9,235,420 (the “’420 Patent), entitled “Branch Target Buffer for Emulation Environments,” was issued by the United States Patent and Trademark Office on January 12, 2016. A true and correct copy of the ’420 Patent is attached as Exhibit 3.

64. The ’420 Patent provides systems and methods for managing indirect branch instructions in an emulation environment. Branch instructions are instructions that are used to change the sequence of execution of program instructions. Generally, the execution of an indirect branch instruction will result in a destination address being calculated that directs the program to the memory location for the next set of instructions to be executed by the program. In an emulation environment where the instructions comprising the program must be translated to a different set of instructions that the processor can execute (translating from one instruction set architecture to another), the repeated calculation of the branch target address in the emulated environment can result in significant unnecessary processing. To avoid unnecessary processing, the ‘420 Patent preserves previously calculated indirect branch target addresses in a buffer, later identifying and using such previously calculated target addresses when they are encountered again. This provides for much more efficient operation in an emulation environment. IBM is the current owner by assignment of all right, title, and interest of the ’420 Patent.

65. U.S. Patent No. 8,713,289 (the “’289 Patent”), entitled “Efficiently Emulating Computer Architecture Condition Code Settings Without Executing Branch Instructions,” was issued by the United States Patent and Trademark Office on April 29, 2014. A true and correct copy of the ’289 Patent is attached as Exhibit 4.

66. The ’289 Patent relates to emulation of computer system architectures. Specifically, the ’289 Patent provides methods and systems for handling condition codes in the emulation process. Condition codes include codes set as the result of execution of an instruction that can indicate an outcome status of the execution, such as an overflow. The ’289 Patent provides sequences of instructions that produce valid condition code settings for an emulated source architecture without the need for branch instructions from the target architecture. This

can help eliminate unnecessary branch instructions in the target code, which helps to improve the efficiency of operation. IBM is the current owner by assignment of all right, title, and interest of the ’289 Patent.

67. United States Patent No. 7,434,209 (the “’209 Patent”), entitled “Method and Apparatus for Performing Native Binding to Execute Native Code,” was issued by the U.S. Patent and Trademark Office on October 7, 2009. A true and correct copy of the ’209 Patent is attached as Exhibit 5.

68. The ’209 Patent is generally directed to a method and apparatus of emulation or other forms of translation that identifies an application call to a service routine native to the environment for which the application was compiled and executes a corresponding service routine native to the environment in which the application is emulated or for which it is otherwise translated. The ’209 Patent provides for such wholesale substitution of a corresponding service routine, rather than merely executing a translated version of the service routine present in the environment for which the application was compiled. The patent provides for much more efficient emulation or other translation of an application and the service routines it calls than to simply emulate or translate the application instructions and the instructions of the service routines it calls. IBM is the current owner by assignment of all right, title, and interest of the ’209 Patent.

LzLabs' Attempts to Sell its SDM in the United States

69. LzLabs offers its SDM for sale to customers within the United States. It does this through general offers and offers for demonstrations on its website. It also offers its SDM for cloud implementation through the Microsoft Azure platform.

70. In a recent press release, LzLabs formally announced expansion into North America to take advantage of the well-established mainframe customer base in North America.[20] The majority of the mainframe customer base in North America is within the United States.

71. In or around February 2021, IBM became aware that LzLabs was communicating with an IBM customer headquartered in Tennessee. Specifically, IBM became aware that LzLabs had scheduled meetings to occur with that customer in Tennessee the week of February 22, 2021, to explore migrating certain of that customer's applications from an IBM mainframe to LzLabs' SDM. The IBM customer invited IBM to this meeting. When the meeting was to begin and it became apparent to LzLabs that IBM was attending, LzLabs canceled the meeting.

72. In connection with those efforts, LzLabs offered to sell its SDM to the IBM customer within the United States and has made a test installation of the SDM in the United States for that IBM customer.

[20] *LzLabs Expands into North American Market, Liberating Legacy Applications*, LzLabs (July 14, 2020), https://www.lzlabs.com/lzlabs-expands-into-north-american-market-liberating-legacy-applications/ (last visited February 17, 2022).

LzLabs' Advertising

73. LzLabs aggressively advertises its SDM, alleging that it has capabilities functionally equivalent to that of IBM's mainframe platform. These statements are false.

74. For example, LzLabs released an "LzLabs Software Defined Mainframe Product Data sheet"[21] ("Data Sheet"). According to its metadata, this pdf was created in October 2018. The Data Sheet states that one "Key Benefit[]" of the SDM is that it is a "[l]ow cost, functionally equivalent platform for existing customer legacy system applications." LzLabs has made similar claims on other occasions, stating that its SDM "supports the necessary functionally-equivalent subsystem APIs to enable transparent execution of the binary representations of these programs and data."[22] LzLabs claims that compiled applications can be migrated from the IBM mainframe environment to the SDM without modification.

75. These statements were false at the time they were made, and remain false today. The SDM is not "functionally equivalent" to IBM's mainframe platform. Rather, LzLabs incrementally implements purportedly equivalent functionality of the IBM mainframe platform on an as-needed basis as required by each new deployment of the SDM. Although LzLabs has implemented certain specific equivalent functionality, it has done so through the misappropriation of IBM's trade secrets and infringement of IBM's patents, as set forth in detail above. But each new deployment of the SDM requires LzLabs to engage in custom development to implement features and functionality of the IBM mainframe platform that LzLabs did not

---

[21] https://query.prod.cms.rt.microsoft.com/cms/api/am/binary/RWBplr (last visited February 17, 2022).

[22] *Swisscom Moves Entire Mainframe Workload to Software Defined Mainframe in the Cloud*, LzLabs, https://www.lzlabs.com/resources/swisscom-moves-entire-mainframe-workload-to-software-defined-mainframe-in-the-cloud/ (last visited February 17, 2022); *LzLabs teams up with Amazon Web Services to deliver legacy mainframe applications in the cloud*, LzLabs, https://www.lzlabs.com/resources/lzlabs-teams-up-with-amazon-web-services-to-deliver-legacy-mainframe-applications-in-the-cloud/ (last visited February 17, 2022).

implement for prior SDM deployments. And each new development requires additional improper reverse engineering of the proprietary IBM software. Therefore, although LzLabs has documented publicly that it has implemented certain specific functionality in its SDM, accomplished through trade secret misappropriation and patent infringement, its general claims of being a functionally equivalent replacement for the IBM mainframe platform are false.[23]

76. These false statements have harmed IBM. These statements misleadingly suggest that LzLabs' SDM is a turnkey mainframe platform replacement when, in reality, each SDM deployment relies on additional improper reverse engineering of IBM's software to add functionality not supported by the SDM. Each new deployment also relies on misappropriation of additional IBM intellectual property. LzLabs never discloses these risks to potential customers, instead claiming that the SDM is a functionally equivalent replacement for the IBM mainframe platform. These false statements mislead current customers of IBM to believe that the SDM is an alternative to the IBM mainframe platform that can operate their existing applications. These false and misleading statements injure IBM by inducing customers to leave the IBM mainframe platform under false pretenses. They also injure IBM by diminishing the reputation of IBM and its mainframe system, suggesting that an offering developed in a short period of time can match the speed, reliability, and security of the IBM mainframe system.

FIRST CAUSE OF ACTION

(Misappropriation of Trade Secrets – Defend Trade Secrets Act, 18 U.S.C. § 1836 *et seq*.)

77. IBM repeats and realleges each and every allegation set forth in paragraphs 1 through 76 as if fully set forth herein.

[23] Even if the SDM could provide a functionally equivalent replacement for a particular IBM mainframe application, which it cannot, the SDM cannot provide the same reliability, security, availability and performance as the IBM mainframe systems.

78. LzLabs' and Texas Wormhole's conduct constitutes a willful and malicious misappropriation of IBM's trade secrets in the United States. Such trade secrets include the structure, function, and operation of IBM's COBOL & PL/I Runtime Service Routines, Middleware Service Routines, and Operating System Services ("Misappropriated Trade Secrets").

79. The Misappropriated Trade Secrets have independent economic value because they are not generally known to, and not readily ascertainable through proper means by, other persons who can obtain economic value from their disclosure or use. For example, the Misappropriated Trade Secrets are not readily ascertainable absent the reverse assembling, reverse compiling, translating, or reverse engineering of the Licensed IBM Software, which IBM distributes with contractual restrictions on such activities.

80. IBM has maintained the secret, confidential information of the Misappropriated Trade Secrets and has taken reasonable measures to keep the information secret, including through employment agreements that require IBM employees to retain such information confidentially and forbid disclosure of such information to anyone outside of IBM. In addition, IBM places contractual restrictions on activities such as reverse assembling, reverse compiling, translating, and reverse engineering of the Licensed IBM Software. Further, IBM source code embodying the Misappropriated Trade Secrets is stored in source code management repositories. Such source code management repositories are accessible only from within the IBM intranet by an authenticated user. They are monitored for unusual network traffic to maintain their security. User access to these source code management repositories is authorized and revalidated on a quarterly basis. IBM employees are also required to take annual cybersecurity education courses.

81. In creating, using, marketing, and selling its SDM, LzLabs and Texas Wormhole misappropriated the Misappropriated Trade Secrets. The SDM incorporates the Misappropriated Trade Secrets, which were derived using improper means. LzLabs and Texas Wormhole had no rights in the Licensed IBM Software, and therefore the reverse engineering, reverse assembling, reverse compiling and/or translating they (or those on their behalf) performed was unauthorized and improper. Although LzLabs set up a shell entity to license the IBM software from IBM UK, LzLabs and Texas Wormhole further knew that any reverse engineering, reverse assembling, reverse compiling and/or translating performed by the shell entity was barred by its agreements with IBM UK.

82. LzLabs has been marketing and offering to sell the fruit of its misappropriation, the SDM, which incorporates or whose development relies on the Misappropriated Trade Secrets. These efforts have taken place within the United States.

83. Texas Wormhole has been using the fruit of its misappropriation, the SDM, which incorporates or whose development relies on, the Misappropriated Trade Secrets. This use has occurred in the United States.

84. LzLabs' and Texas Wormhole's misappropriation of IBM's Misappropriated Trade Secrets was willful.

85. IBM has been damaged as a result of LzLabs' and Texas Wormhole's conduct, and seeks damages in accordance with proof at trial, but in any event sufficient to: (1) compensate it for its actual losses, including lost profits resulting from LzLabs' and Texas Wormhole's misappropriation, and (2) recover the amounts that LzLabs and Texas Wormhole unjustly received as a result of its misappropriation of the Misappropriated Trade Secrets. In lieu

of the above, IBM is entitled to a reasonable royalty for LzLabs' and Texas Wormhole's misappropriation.

86. In addition, because LzLabs' and Texas Wormhole's misappropriation was willful and malicious, IBM is entitled to recover exemplary damages in an amount equal to twice the damages otherwise recoverable, and to recover its attorneys' fees and costs of suit.

87. If Defendants are not enjoined Defendants will continue to misappropriate and use IBM's trade secrets for their own benefit and to IBM's detriment.

SECOND CAUSE OF ACTION
(Misappropriation of Trade Secrets – Texas Uniform Trade Secrets Act)

88. IBM repeats and realleges each and every allegation set forth in paragraphs 1 through 87 as if fully set forth herein.

89. LzLabs' and Texas Wormhole's conduct constitutes a willful and malicious misappropriation, within this state, of the Misappropriated Trade Secrets to which IBM holds exclusive rights in the United States.

90. The Misappropriated Trade Secrets have independent economic value because they are not generally known to, and not readily ascertainable through proper means by, other persons who can obtain economic value from their disclosure or use. For example, the Misappropriated Trade Secrets are not readily ascertainable absent the reverse engineering, reverse assembling, reverse compiling and/or translating of the Licensed IBM Software, which IBM only distributes with contractual restrictions on such activities.

91. IBM has maintained the secret, confidential information of the Misappropriated Trade Secrets and has taken reasonable measures to keep the information secret, including through employment agreements that require IBM employees to retain such information confidentially and through contractual restrictions on activities such as reverse

assembling, reverse compiling, translating, and reverse engineering of the Licensed IBM Software.

92. In creating, using, marketing, and selling its SDM, LzLabs and Texas Wormhole misappropriated the Misappropriated Trade Secrets. The SDM incorporates the Misappropriated Trade Secrets, which were derived using improper means. LzLabs and Texas Wormhole had no rights in the Licensed IBM Software, and therefore the reverse engineering, reverse assembling, reverse compiling and/or translating they performed was unauthorized and improper. Although LzLabs set up a shell entity to license the IBM software from IBM UK, LzLabs and Texas Wormhole further knew that any reverse assembling, reverse compiling, translating, or reverse engineering performed by the shell entity was barred by its agreements with IBM UK.

93. LzLabs' and Texas Wormhole's misappropriation occurred at least in part in the state of Texas. All of Texas Wormhole's operations occur in Texas. In addition, LzLabs' employees within Texas, including Steve Towns, Gary Trinklein, Tom Harper, and Tommy Sprinkle engaged in misappropriation within the state of Texas through, at least, their use of the SDM within the state.

94. LzLabs' and Texas Wormhole's misappropriation of IBM's Misappropriated Trade Secrets was willful.

95. IBM has been damaged by LzLabs' and Texas Wormhole's conduct, and seeks damages in accordance with proof at trial, but in any event sufficient to: (1) compensate it for its actual losses, including lost profits resulting from LzLabs' and Texas Wormhole's misappropriation, and (2) recover the amounts that LzLabs and Texas Wormhole unjustly received as a result of its misappropriation of the Misappropriated Trade Secrets. In lieu of the

above, IBM is entitled to a reasonable royalty for LzLabs' and Texas Wormhole's misappropriation.

96. In addition, because LzLabs' and Texas Wormhole's misappropriation was willful and malicious, IBM is entitled to recover exemplary damages in an amount equal to twice the damages otherwise recoverable, and to recover its attorneys' fees and costs of suit.

97. If Defendants are not enjoined Defendants will continue to misappropriate and use IBM's trade secrets for their own benefit and to IBM's detriment.

THIRD CAUSE OF ACTION
('823 Patent Infringement – 35 U.S.C. § 271)

98. IBM repeats and realleges each and every allegation set forth in paragraphs 1 through 97 as if fully set forth herein.

99. Claim 1 of the '823 Patent is exemplary, and states as follows:

> A method for executing a machine instruction in a central processing unit, the method comprising:
>
> obtaining, by a processor, a machine instruction for execution, the machine instruction being defined for computer execution according to a computer architecture, the machine instruction comprising an opcode, one register field, another register field, an index field, a base field, and a displacement value;
>
> performing a shift function on a significand of a decimal floating point datum as that function is defined by the opcode of the machine instruction, wherein the significand is stored in a location designated by the one register field, the shift function comprising shifting in one direction one or more decimal digits of the significand a number of positions specified by a plurality of second operand bits determined using the index field, the base field and displacement value of the machine instruction; and
>
> placing a result of the shift function in a location designated by the other register field.

100. LzLabs' SDM product, used by Texas Wormhole, meets all the limitations of at least claim 1 of the '823 Patent in violation of 35 U.S.C. § 271(a).

101. LzLabs and Texas Wormhole have directly infringed, continue to infringe, and/or, at least as of the filing of this Complaint, induce or contribute to the infringement by others of one or more claims of the ’823 Patent by making, using, selling, offering for sale, and/or importing into the United States, without authority of license, the “Software Defined Mainframe” or “SDM” in violation of 35 U.S.C. §§ 271(a), (b), and (c). For example, LzLabs has offered to sell its SDM to an IBM customer in Tennessee. LzLabs has also implemented the SDM on Microsoft’s Azure cloud services in the United States. Texas Wormhole has used the SDM in the United States in the course of its SDM development within the United States, and within Texas.

102. By at least the filing of this Complaint, IBM has disclosed the existence of the ’823 Patent and identified at least some of LzLabs’ and Texas Wormhole’s activities that infringe at least one claim of the ’823 Patent. Thus, based on this disclosure, LzLabs and Texas Wormhole have knowledge of the ’823 Patent and that their activities infringe the ’823 Patent. Based on IBM’s disclosures, LzLabs and Texas Wormhole have also known or should have known since at least the filing of this Complaint that customers, distributors, suppliers, and other purchasers of the SDM product are infringing the ’823 Patent at least because LzLabs and Texas Wormhole have known that they are infringing the ’823 Patent.

103. For example, the SDM provides support for IBM COBOL Version 5 (COBOL V5) as described above. *See The Anatomy of Mainframe Workload Migration*, LzLabs (Apr. 2018) (https://www.lzlabs.com/resources/the-anatomy-of-mainframe-application-workload) (last visited February 17, 2022). Through this support, the SDM implements each of the IBM mainframe z/Architecture instructions that make up a COBOL application program compiled for execution on the IBM mainframe, and, in order to execute each such instruction, translates it

(singly or as part of a group of instructions encompassing it) into one or more x86 architecture instructions that collectively produce the same results. For application programs compiled with COBOL V5 with the ARCH(10 or 11) option, this translation includes the Shift Significand Left and Shift Significand Right instructions that are the subject of the '823 patent. *See* the '823 Patent at 26:15-55.

104. Because application programs compiled with COBOL V5 with the ARCH(10 or 11) option can include these "Shift Significand" instructions, the LzLabs translation must thereby include the translation of those instructions, which are defined for computer execution according to the z/Architecture. These "Shift Significand" instructions each have an opcode (the combination of the values at bits 0-7 and 40-47); one register field (R_3); another register field (R_1); an index field (X_2); a base field (B_2); and a displacement value (D_2). *Id.*

105. Further, as defined by the z/Architecture, the opcodes for these instructions specify a left or right shift of the significand of a decimal floating point number. In order to meet its claim of obtaining the same results when "executing" the object form of an application compiled for the IBM mainframe, the LzLabs methodology must translate the application mainframe instructions into x86 instructions that perform the same significand shift operation on that same decimal floating point number. The location of the significand, the number of digits to shift, and the location where the results are to be placed are each specified by the z/Architecture, and the LzLabs SDM must adhere to that specification.

106. Specifically, as defined by the z/Architecture and, as performed by the SDM according to LzLabs' specific functionality claims, the significand to be shifted left or right is located in the register, or register pair, designated by the one register field (R_3). Then, the result of the shift operation is placed in the register or register pair, designated by the other register

field (R_1). The number of bits to be shifted is specified by the second operand of the instruction, *i.e.*, the rightmost 6 bits of the sum of the values in the registers specified in the index (X_2) and base (B_2) fields and the value specified in the displacement field (D_2).

107. At least as of the date of the filing of this Complaint and based on the information set forth herein, LzLabs also actively, knowingly, and intentionally induces infringement of one or more claims of the ’823 Patent under 35 U.S.C. § 271(b) by actively encouraging others to import, make, use, sell, and/or offer to sell in the United States, the SDM product. LzLabs instructs its customers how to use the SDM product by allowing customer applications written and compiled for the IBM mainframe environment, and utilizing proprietary IBM software services, including the IBM Middleware Service Routines, IBM Operating System Service Routines, and IBM COBOL & PL/I Service Routines, “without changes and without compromise” on the SDM product, as described above. For example, the LzLabs Software Designed Mainframe Product Data Sheet states the SDM enables applications written in COBOL and PL/I and offers replacements for CICS and IMS.[24]

108. At least as of the date of the filing of this Complaint and based on the information set forth herein, LzLabs further contributes to the infringement of one more claims of the ’823 Patent under 35 U.S.C. § 271(c) by offering to sell, selling, and/or importing into the United States a component of the SDM product, or a material or apparatus for using in practicing a process of claims in the ’823 Patent, that constitutes a material part of the inventions, knowing the same to be especially made or especially adapted for use in an infringement of the ’823 Patent, and is not a staple article or commodity of commerce suitable for substantial

[24] https://query.prod.cms.rt.microsoft.com/cms/api/am/binary/RWBplr (last visited February 17, 2022).

noninfringing use. In this case, LzLabs' SDM product is a material part of at least the invention of claim 1 of the '823 Patent for the reasons set forth herein.

109. LzLabs' and Texas Wormhole's patent infringement is willful, at least because the SDM was created by copying the functionality of IBM hardware.

110. LzLabs' and Texas Wormhole's infringement has damaged and continues to damage IBM in an amount yet to be determined, of at least a reasonable royalty and/or the lost profits that IBM would have made but for LzLabs' acts of infringement.

FOURTH CAUSE OF ACTION
('664 Patent Infringement – 35 U.S.C. § 271)

111. IBM repeats and realleges each and every allegation set forth in paragraphs 1 through 110 as if fully set forth herein.

112. Claim 14 of the '664 Patent is exemplary, and states as follows:

> A method of executing a machine instruction by a central processing unit, said method comprising: executing a machine instruction, the machine instruction being defined for computer execution according to a computer architecture and comprising an opcode and a mask field, the executing comprising: performing an operation that provides a result, said result comprising a sign; and indicating by a selection indicator of the mask field how the sign is to be encoded in response to the sign being a specified sign.

113. LzLabs' SDM product, used by Texas Wormhole, meets all the limitations of at least claim 14 of the '664 Patent in violation of 35 U.S.C. § 271(a).

114. LzLabs and Texas Wormhole have directly infringed, continue to infringe, and/or, at least as of the filing of this Complaint, induce or contribute to the infringement by others of one or more claims of the '664 Patent by making, using, selling, offering for sale, and/or importing into the United States, without authority of license, the "Software Defined Mainframe" or "SDM" in violation of 35 U.S.C. §§ 271(a), (b), and (c). For example, LzLabs has offered to sell its SDM to an IBM customer in Tennessee. LzLabs has also implemented the

SDM on Microsoft's Azure cloud services in the United States. Texas Wormhole has used the SDM in the United States in its SDM development within the United States, and within Texas.

115. By at least the filing of this Complaint, IBM has disclosed the existence of the '664 Patent and identified at least some of LzLabs' and Texas Wormhole's activities that infringe at least one claim of the '664 Patent. Thus, based on this disclosure, LzLabs and Texas Wormhole have knowledge of the '664 Patent and that their activities infringe the '664 Patent. Based on IBM's disclosures, LzLabs and Texas Wormhole have also known or should have known since at least the filing of this Complaint that customers, distributors, suppliers, and other purchasers of the SDM product are infringing the '664 Patent at least because LzLabs and Texas Wormhole have known that they are infringing the '664 Patent.

116. For example, LzLabs claims that the SDM provides support for IBM COBOL Version 5 (COBOL V5).[25] Through this support, the SDM implements each of the IBM mainframe z/Architecture instructions that make up a COBOL application program compiled for execution on the IBM mainframe, and in order to execute each such instruction, translates it into one or more x86 architecture instructions that collectively accomplish the same results.

117. Because application programs compiled with COBOL V5 with the ARCH(8, 9, 10, or 11) option can, and typically do, include "Convert to Signed Packed" instructions, the LzLabs translation must thereby include the translation of those instructions, which are defined for computer execution according to the z/Architecture. These "Convert to Signed Packed" instructions include both an opcode and a mask field. *See* the '644 Patent at 25:29-26:11.

[25] See *The Anatomy of Mainframe Workload Migration*, LzLabs (Apr. 2018) (https://www.lzlabs.com/resources/the-anatomy-of-mainframe-application-workload/) (last visited February 17, 2022).

118. Lastly, when performing these machine instructions by a central processing unit, the SDM performs an operation that provides a result, said result comprising a sign; and indicating by a selection indicator of the mask field how the sign is to be encoded in response to the sign being a specified sign. Specifically, as defined by the z/Architecture, the Convert to Signed Packed instructions operate to convert a decimal floating point number to a packed decimal number. Because the encoding for the sign of packed decimal numbers can vary, the z/Architecture further specifies that the particular encoding of the sign portion of the result is based on a selection indicator (bit 3) in the Mask field (M_4) of the Convert to Signed Packed instruction. *Id.* Thus, the generated instructions by the SDM will necessarily utilize this indicator in the Mask field to achieve a properly encoded sign result, which is necessary to meet LzLabs' claims of results equivalent to those achieved when the application runs on the IBM mainframe.

119. At least as of the date of the filing of this Complaint and based on the information set forth herein, LzLabs also actively, knowingly, and intentionally induces infringement of one or more claims of the '664 Patent under 35 U.S.C. § 271(b) by actively encouraging others to import, make, use, sell, and/or offer to sell in the United States, the SDM product. LzLabs instructs its customers how to use the SDM product by allowing customer applications written and compiled for the IBM mainframe environment, and utilizing proprietary IBM software services, including the IBM Middleware Service Routines, Operating System Service Routines, and IBM COBOL & PL/I Service Routines, "without changes and without compromise" on the SDM product, as described above. For example, the LzLabs Software

Designed Mainframe Product Data Sheet states the SDM enables applications written in COBOL and PL/I and offers replacements for CICS and IMS.[26]

120. At least as of the date of the filing of this Complaint and based on the information set forth herein, LzLabs further contributes to the infringement of one more claims of the ’664 Patent under 35 U.S.C. § 271(c) by offering to sell, selling, and/or importing into the United States a component of the SDM product, or a material or apparatus for using in practicing a process of claims in the ’664 Patent, that constitutes a material part of the inventions, knowing the same to be especially made or especially adapted for use in an infringement of the ’664 Patent, and is not a staple article or commodity of commerce suitable for substantial noninfringing use. In this case, LzLabs’ SDM product is a material part of at least the invention of claim 14 of the ’664 Patent for the reasons set forth herein.

121. LzLabs’ and Texas Wormhole’s patent infringement is willful, at least because the SDM was created by copying the functionality of IBM hardware.

122. LzLabs’ and Texas Wormhole’s infringement has damaged and continues to damage IBM in an amount yet to be determined, of at least a reasonable royalty and/or the lost profits that IBM would have made but for LzLabs’ acts of infringement.

FIFTH CAUSE OF ACTION
(’420 Patent Infringement – 35 U.S.C. § 271)

123. IBM repeats and realleges each and every allegation set forth in paragraphs 1 through 122 as if fully set forth herein.

124. Claim 1 of the ’420 Patent is exemplary, and states as follows:

> A method for managing branch instructions in an emulation environment that is executing a program, the method comprising:

[26] https://query.prod.cms.rt.microsoft.com/cms/api/am/binary/RWBplr (last visited February 17, 2022).

> populating a plurality of entries in a branch target buffer residing within an emulated environment in which the program is executing, each of the entries comprising an instruction address and a target address of a branch instruction of the program;
>
> based on an indirect branch instruction of the program being encountered,
>
> obtaining an indirect branch instruction target key for the indirect branch instruction;
>
> analyzing, by a processor based on the indirect branch instruction target key, one of the entries in the branch target buffer to determine if the instruction address of the one entry is associated with a target address of the indirect branch instruction, wherein the analyzing comprises comparing the instruction address of the one entry to the indirect branch instruction target key; and
>
> based on the instruction address of the one entry being associated with the target address of the indirect branch instruction, branching to the target address of the one entry.

125. LzLabs' SDM product, used by Texas Wormhole, meets all the limitations of at least claim 1 of the '420 Patent in violation of 35 U.S.C. § 271(a).

126. LzLabs and Texas Wormhole have directly infringed, continue to infringe, and/or, at least as of the filing of this Complaint, induce or contribute to the infringement by others of one or more claims of the '420 Patent by making, using, selling, offering for sale, and/or importing into the United States, without authority of license, the "Software Defined Mainframe" or "SDM" in violation of 35 U.S.C. §§ 271(a), (b), and (c). For example, LzLabs has offered to sell its SDM to an IBM customer in Tennessee. LzLabs has also implemented the SDM on Microsoft's Azure cloud services in the United States. Texas Wormhole has used the SDM in the United States in its SDM development within the United States, and within Texas.

127. By at least the filing of this Complaint, IBM has disclosed the existence of the '420 Patent and identified at least some of LzLabs' and Texas Wormhole's activities that infringe at least one claim of the '420 Patent. Thus, based on this disclosure, LzLabs and Texas

Wormhole have knowledge of the ’420 Patent and that their activities infringe the ’420 Patent. Based on IBM’s disclosures, LzLabs and Texas Wormhole have also known or should have known since at least the filing of this Complaint that customers, distributors, suppliers, and other purchasers of the SDM product are infringing the ’420 Patent at least because LzLabs and Texas Wormhole have known that they are infringing the ’420 Patent.

128. For example, as described above, the SDM emulates computer instructions originating from a source machine, such as the IBM mainframe, to produce sequences of instructions on a target machine. This emulation allows a customer to utilize IBM services “without changes and without compromise” as described above. In addition, the SDM has stated design objectives that it will provide emulation/translation services “with behavioral equivalence,” and that “[s]uch translation involves much more than simply converting one instruction set into another, memory management must be preserved plus a range of other considerations taken to ensure functional compatibility. But the end result” of the emulation “behaves exactly like the original mainframe application program.” *The Software Defined Mainframe – Leveraging “the Power of Modern*”, LzLabs (Sept. 2017).

129. As described in the ’420 Patent, the branch target buffer entries preserve the actual memory location, or “target address,” of an indirect branch instruction, when that location is first determined upon encountering an indirect branch instruction whose calculated target “instruction address” is processed to determine the actual location (target address) of the code where execution is to continue as a result of executing the indirect branch instruction. The branch target buffer functions as a cache, allowing faster processing of an indirect branch instruction whenever the calculated target “instruction address” of that instruction has an entry in the buffer. *See* the ’420 Patent at 5:24-6:21.

130. To meet its stated performance objectives, the SDM creates and populates, as the mainframe application load module runs, a multi-entry buffer in its emulation environment, each of the entries comprising an instruction address and a target address of a branch instruction of the program.

131. Many indirect branches are typically present and often executed repeatedly in COBOL and PL/I mainframe applications. LzLabs' claims of equivalent performance to that of the IBM mainframe cannot be credible if the LzLabs SDM does not implement a branch target buffer to reduce the performance penalty otherwise associated with determining, each time an indirect branch instruction is encountered, the actual memory location of the code where execution is to continue as a result of executing the indirect branch instruction. LzLabs makes use of such a branch target buffer as it has stated that one mode of the SDM "'remembers' the branch target address so as to avoid returning to the [incremental compiler] the next time that branch is executed" and that this mode "can reduce application elapsed times by 40% when compared to the same application running in interpretive execution mode."[27]

132. Lastly, the SDM uses the indirect branch target instruction address, which is a memory address in the program being emulated, as a key to search for a matching key to determine if the actual location of the code, which is the corresponding memory address in the emulation environment for the program being emulated, where execution is to continue (the target address) has previously been determined and used to populate an entry in the branch target buffer, and if so, to identify that entry. *See id.* If an entry in the branch target buffer with a

[27] *The Anatomy of Mainframe Workload Migration*, LzLabs (Apr. 2018) (https://www.lzlabs.com/resources/the-anatomy-of-mainframe-application-workload) (last visited February 17, 2022).

matching key is found, the SDM uses the target address stored in that entry as the actual target address for the current indirect branch instruction and branches to that address. *See id.*

133. At least as of the date of the filing of this Complaint and based on the information set forth herein, LzLabs also actively, knowingly, and intentionally induces infringement of one or more claims of the '420 Patent under 35 U.S.C. § 271(b) by actively encouraging others to import, make, use, sell, and/or offer to sell in the United States, the SDM product. LzLabs instructs its customers how to use the SDM product by allowing customer applications written and compiled for the IBM mainframe environment, and utilizing proprietary IBM software services, including the IBM Middleware Service Routines, IBM Operating System Service Routines, and IBM COBOL & PL/I Service Routines, "without changes and without compromise" on the SDM product, as described above. For example, the LzLabs Software Designed Mainframe Product Data Sheet states the SDM enables applications written in COBOL and PL/I and offers replacements for CICS and IMS.[28]

134. At least as of the date of the filing of this Complaint and based on the information set forth herein, LzLabs further contributes to the infringement of one more claims of the '420 Patent under 35 U.S.C. § 271(c) by offering to sell, selling, and/or importing into the United States a component of the SDM product, or a material or apparatus for using in practicing a process of claims in the '420 Patent, that constitutes a material part of the inventions, knowing the same to be especially made or especially adapted for use in an infringement of the '420 Patent, and is not a staple article or commodity of commerce suitable for substantial noninfringing use. In this case, LzLabs' SDM product is a material part of at least the invention of claim 1 of the '420 Patent for the reasons set forth herein.

[28] https://query.prod.cms.rt.microsoft.com/cms/api/am/binary/RWBplr (last visited February 17, 2022).

135. LzLabs' and Texas Wormhole's patent infringement is willful, at least because the SDM was created by copying the functionality of proprietary IBM software services.

136. LzLabs' and Texas Wormhole's infringement has damaged and continues to damage IBM in an amount yet to be determined, of at least a reasonable royalty and/or the lost profits that IBM would have made but for LzLabs' acts of infringement.

SIXTH CAUSE OF ACTION
('289 Patent Infringement – 35 U.S.C. § 271)

137. IBM repeats and realleges each and every allegation set forth in paragraphs 1 through 136 as if fully set forth herein.

138. Claim 16 of the '289 Patent is exemplary, and states as follows:

> A method for emulating computer instructions from a source machine to produce sequences of instructions on a target machine, said method comprising:
>
> obtaining by the target machine a computer instruction from the source machine, said source machine having a different architecture from said target machine; and
>
> generating a sequence of target machine instructions which together operate to derive an encoding for a target machine condition code for the computer instruction, wherein the sequence of target machine instructions provides distinguishing information to distinguish between a plurality of possible outcomes for the target machine condition code, and directly calculates the target machine condition code without using branch instructions, the directly calculating comprising:
>
> determining an intermediate condition code value, the intermediate condition code value being a provisional value for the target machine condition code and subject to change based on the distinguishing information; and
>
> determining, based on the intermediate condition code value and based on the distinguishing information, the target machine condition code, wherein at least part of said distinguishing information is separate from the intermediate condition code value.

139. LzLabs' SDM product, used by Texas Wormhole, meets all the limitations of at least claim 16 of the '289 Patent in violation of 35 U.S.C. § 271(a).

140. LzLabs and Texas Wormhole have directly infringed, continue to infringe, and/or, at least as of the filing of this Complaint, induce or contribute to the infringement by others of one or more claims of the '289 Patent by making, using, selling, offering for sale, and/or importing into the United States, without authority of license, the "Software Defined Mainframe" or "SDM" in violation of 35 U.S.C. §§ 271(a), (b), and (c). For example, LzLabs has offered to sell its SDM to an IBM customer in Tennessee. LzLabs has also implemented the SDM on Microsoft's Azure cloud services in the United States. Texas Wormhole has used the SDM in the United States in the course of its SDM development within the United States, and within Texas.

141. By at least the filing of this Complaint, IBM has disclosed the existence of the '289 Patent and identified at least some of LzLabs' and Texas Wormhole's activities that infringe at least one claim of the '289 Patent. Thus, based on this disclosure, LzLabs and Texas Wormhole have knowledge of the '289 Patent and that their activities infringe the '289 Patent. Based on IBM's disclosures, LzLabs and Texas Wormhole have also known or should have known since at least the filing of this Complaint that customers, distributors, suppliers, and other purchasers of the SDM product are infringing the '289 Patent at least because LzLabs and Texas Wormhole have known that they are infringing the '289 Patent.

142. For example, as described above, the SDM emulates computer instructions originating from a source machine to produce sequences of instructions on a target machine. This emulation allows a customer to utilize IBM services "without changes and without compromise" as described above. In addition, the SDM has stated design objectives that it will

provide emulation services "with behavioral equivalence," to that of IBM's services and that "the end result" of the emulation "behaves exactly like the original mainframe application program." *The Software Defined Mainframe – Leveraging "the Power of Modern*", LzLabs (Sept. 2017).

143. To perform this emulation, the SDM first obtains, by the "target machine" with an x86 architecture, a computer instruction that is intended for a "source machine" having a z/Architecture. *Id.* Once it receives this instruction, the SDM generates a sequence of target instructions which together operate to derive an IBM mainframe condition code representation on the target machine, the "target machine condition code," for a particular IBM mainframe source instruction, as such a condition code may be required to execute a subsequent IBM mainframe source instruction. Performance considerations for a processor, such as an x86 machine using pipelining, dictates that this be done without using branch instructions. Additionally, target instructions would provide information distinguishing between different IBM mainframe condition codes to ensure that the correct condition code is generated. Because IBM mainframe instructions that set the condition code are common in COBOL and PL/I compile programs, if the "branchless" calculations of the '289 patent were not used, efficiency and performance of related services would noticeably decrease. Thus, because LzLabs claims that there is no such performance degradation, its SDM must make use of these "branchless" calculations when determining the claimed condition code.

144. Lastly, when generating these condition codes without the use of branch instructions, the SDM makes use of intermediate condition code values that are updated throughout the calculation process. For example, the procedure for generating a condition code without branches involves successive updates to an IBM mainframe condition code representation on the target machine based on other values such as sign bits and the like

("distinguishing information"). Values stored in the mainframe condition code representation before the final update are an "intermediate" condition code value subject to change based on "distinguishing information" used for final update and determination of the condition code.

145. At least as of the date of the filing of this Complaint and based on the information set forth herein, LzLabs also actively, knowingly, and intentionally induces infringement of one or more claims of the '289 Patent under 35 U.S.C. § 271(b) by actively encouraging others to import, make, use, sell, and/or offer to sell in the United States, the SDM product. LzLabs instructs its customers how to use the SDM product by allowing customer applications written and compiled for the IBM mainframe environment, and utilizing proprietary IBM software services, including the IBM Middleware Service Routines, Operating System Service, Routines, and IBM COBOL & PL/I Service Routines, "without changes and without compromise" on the SDM product, as described above. For example, the LzLabs Software Designed Mainframe Product Data Sheet states the SDM enables applications written in COBOL and PL/I and offers replacements for CICS and IMS.[29]

146. At least as of the date of the filing of this Complaint and based on the information set forth herein, LzLabs further contributes to the infringement of one more claims of the '289 Patent under 35 U.S.C. § 271(c) by offering to sell, selling, and/or importing into the United States a component of the SDM product, or a material or apparatus for using in practicing a process of claims in the '289 Patent, that constitutes a material part of the inventions, knowing the same to be especially made or especially adapted for use in an infringement of the '289 Patent, and is not a staple article or commodity of commerce suitable for substantial

[29] https://query.prod.cms.rt.microsoft.com/cms/api/am/binary/RWBplr (last visited February 17, 2022).

noninfringing use. In this case, LzLabs' SDM product is a material part of at least the invention of claim 16 of the '289 Patent for the reasons set forth herein.

147. LzLabs' and Texas Wormhole's patent infringement is willful, at least because the SDM was created by copying the functionality of IBM hardware.

148. LzLabs' and Texas Wormhole's infringement has damaged and continues to damage IBM in an amount yet to be determined, of at least a reasonable royalty and/or the lost profits that IBM would have made but for LzLabs' acts of infringement.

SEVENTH CAUSE OF ACTION
('209 Patent Infringement – 35 U.S.C. § 271)

149. IBM repeats and realleges each and every allegation set forth in paragraphs 1 through 148 as if fully set forth herein.

150. Claim 1 of the '209 Patent is exemplary, and states as follows:

> A method of translating subject program code executable by a subject processor into target program code executable by a target processor, said method comprising:
>
> dividing the subject program code into a plurality of subject program code units;
>
> translating one or more of the subject program code units into one or more target program code units; and executing the one or more target program code units on the target processor;
>
> wherein the translating step includes identifying a subject function in the subject program code having a corresponding native function of native code, wherein the native code is code executable by the target processor, and identifying the native function of the native code which corresponds to the identified subject function; and
>
> wherein the executing step includes executing the native function on the target processor instead of executing a translated version of the identified subject function, including transforming zero or more function parameters from a target code representation to a native code representation, invoking the native function with the transformed zero or more functions parameters according to a

> prototype of the native function, and transforming zero or more return values of the invoked native function form a native code representation to a target code representation.

151. LzLabs' SDM product, used by Texas Wormhole, meets all of the limitations of at least claim 1 of the '209 Patent in violation of 35 U.S.C. § 271(a).

152. LzLabs and Texas Wormhole have directly infringed, continue to infringe, and/or, at least as of the filing of this Complaint, induce or contribute to the infringement by others of one or more claims of the '209 Patent by making, using, selling, offering for sale, and/or importing into the United States, without authority of license, the "Software Defined Mainframe" or "SDM" in violation of 35 U.S.C. §§ 271(a), (b), and (c). For example, LzLabs has offered to sell its SDM to an IBM customer in Tennessee. LzLabs has also implemented the SDM on Microsoft's Azure cloud services in the United States.[30] Texas Wormhole has used the SDM in the United States in the course of its SDM development within the United States, and within Texas.

153. By at least the filing of this Complaint, IBM has disclosed the existence of the '209 Patent and identified at least some of LzLabs' and Texas Wormhole's activities that infringe at least one claim of the '209 Patent. Thus, based on this disclosure, LzLabs and Texas Wormhole have knowledge of the '209 Patent and that their activities infringe the '209 Patent. Based on IBM's disclosures, LzLabs and Texas Wormhole have also known or should have known since at least the filing of this Complaint that customers, distributors, suppliers, and other purchasers of the SDM product are infringing the '209 Patent at least because LzLabs and Texas Wormhole have known that they are infringing the '209 Patent.

---

[30] Windows Application Page for LzLabs' Software Defined Mainframe, LzLabs, https://appsource.microsoft.com/en-us/product/web-apps/lzlabsgmbh-5083555.lzlabs-softwaredefinedmainframe (last visited February 17, 2022).

154. For example, the SDM translates subject program code executable by a subject processor into target program code executable by a "target processor" by effectively duplicating the IBM mainframe and software environment on an x86 platform. The SDM divides the subject program code into a plurality of subject program code units by separately processing each of the object or load modules that make up a mainframe application. *The Software Defined Mainframe – Leveraging "the Power of Modern*", LzLabs (Sept. 2017). The SDM parses IBM mainframe code into individual instructions or sequences prior to translating them into equivalent x86 instructions. The SDM translates one or more of the subject program code units into one or more target program code units. The SDM must translate the IBM mainframe Instruction Set Architecture code into x86 instructions for them to execute on the SDM x86 platform. *Id*. The SDM executes one or more target program code units on the target processor when they are executed on the x86 platform.

155. Further, when translating IBM program code, the SDM identifies a subject function in the subject program code having a corresponding native function of native code, wherein the native code is executable by the target processor. As part of the overall translation process, the SDM identifies IBM mainframe application calls to IBM run-time services, and native x86 services are substituted for IBM mainframe services. Further, the SDM identifies the native function of the native code that corresponds to the identified subject function. According to LzLabs, "[w]hen legacy application programs are placed into the container, the customers' programs are enhanced to run on modern computers and decades-old APIs are exchanged for newer, more contemporary ones." *LzLabs appoints Gartner's Dale Vecchio as Chief Marketing Officer*, LzLabs (Mar. 17, 2017), https://www.lzlabs.com/resources/lzlabs-appoints-gartners-dale-vecchio-chief-marketing-officer/ (last visited February 17, 2022).

156. Lastly, when executing the target program code on the target processor, the SDM executes the native function on the target processor instead of executing a translated version of the identified subject function, including transforming zero or more function parameters from a target code representation to a native code representation, invoking the native function with the transformed zero or more function parameters according to a prototype of the native function, and transforming zero or more return values of the invoked native function from a native code representation to a target code representation. The SDM must execute the native function corresponding to the IBM run-time service because there are no IBM run-time services native to the SDM environment. According to LzLabs, the "SDM is a managed customer application container technology that provides the capabilities for mainframe applications to execute on open systems, with no requirement for recompilation or conversion of data types."[31] The SDM's use of native APIs requires parameter transformations to permit handling by such APIs. Additionally, as explained by LzLabs, "[t]he general objective at LzLabs is to develop the SDM as the smallest possible software layer to map mainframe applications APIs and needs to the equivalent Linux APIs." Didier Durand & Dale Vecchio, *As Much Mainframe as Needed, But as Little as Possible*, LzLabs (June 21, 2017), https://www.linkedin.com/pulse/much-mainframe-needed-little-possible-didier-durand/ (last visited February 17, 2022).

157. At least as of the date of the filing of this Complaint and based on the information set forth herein, LzLabs also actively, knowingly, and intentionally induces infringement of one or more claims of the '209 Patent under 35 U.S.C. § 271(b) by actively encouraging others to import, make, use, sell, and/or offer to sell in the United States, the SDM

[31] https://www.lzlabs.com/resources/lzlabs-teams-up-with-amazon-web-services-to-deliver-legacy-mainframe-applications-in-the-cloud/ (last visited February 17, 2022)

product. LzLabs instructs its customers how to use the SDM product by allowing customer applications written and compiled for the IBM mainframe environment, and utilizing proprietary IBM software services, including the IBM Middleware Service Routines, Operating System Service Routines and IBM COBOL & PL/I Service Routines, "without changes and without compromise" on the SDM product, as described above. For example, the LzLabs Software Designed Mainframe Product Data Sheet states the SDM enables applications written in COBOL and PL/I and offers replacements for CICS and IMS.[32]

158. At least as of the date of the filing of this Complaint and based on the information set forth herein, LzLabs further contributes to the infringement of one more claims of the '209 Patent under 35 U.S.C. § 271(c) by offering to sell, selling, and/or importing into the United States a component of the SDM product, or a material or apparatus for using in practicing a process of claims in the '209 Patent, that constitutes a material part of the inventions, knowing the same to be especially made or especially adapted for use in an infringement of the '209 Patent, and is not a staple article or commodity of commerce suitable for substantial noninfringing use. In this case, LzLabs' SDM product is a material part of at least the invention of claim 1 of the '209 Patent for the reasons set forth herein.

159. LzLabs' and Texas Wormhole's patent infringement is willful, at least because the SDM was created by copying the functionality of IBM software.

160. LzLabs' and Texas Wormhole's infringement has damaged and continues to damage IBM in an amount yet to be determined, of at least a reasonable royalty and/or the lost profits that IBM would have made but for LzLabs' acts of infringement.

[32] https://query.prod.cms.rt.microsoft.com/cms/api/am/binary/RWBplr (last visited February 17, 2022).

EIGHTH CAUSE OF ACTION
(False Advertising – Lanham Act 15 U.S.C. § 1125(a))

161. IBM repeats and realleges each and every allegation set forth in paragraphs 1 through 160 as if fully set forth herein.

162. LzLabs' advertisements and promotions for its SDM contain false statements of fact intended to deceive and mislead customers by misrepresenting the SDM's functionality and its compatibility with customer applications, which have been designed and compiled to work with IBM's mainframe.

163. For example, LzLabs' Software Defined Mainframe Product Data Sheet states that the SDM is a "[l]ow cost, functionally equivalent platform for existing customer legacy system applications." LzLabs has made a similar claim several other times, claiming that its SDM "supports the necessary functionally-equivalent subsystem APIs to enable transparent execution of the binary representations of these programs and data."[33] These statements are both literally false and misleading.

164. The statements are literally, objectively false in that the SDM does not offer the range of functions that IBM's mainframe offers to customers—a fact which can be objectively verified by a comparison of the functions available on each platform. Thus, the SDM cannot be a functionally equivalent platform.

165. The statements are also misleading because they indicate to customers that they can use their existing IBM-compatible application with no modification to or interruption in

[33] *Swisscom Moves Entire Mainframe Workload to Software Defined Mainframe in the Cloud*, LzLabs, https://www.lzlabs.com/resources/swisscom-moves-entire-mainframe-workload-to-software-defined-mainframe-in-the-cloud/ (last visited February 17, 2022); *LzLabs teams up with Amazon Web Services to deliver legacy mainframe applications in the cloud*, LzLabs, https://www.lzlabs.com/resources/lzlabs-teams-up-with-amazon-web-services-to-deliver-legacy-mainframe-applications-in-the-cloud/ (last visited February 17, 2022)

the current operation and performance of those applications. Each new deployment of the SDM requires LzLabs to undertake additional custom development implementing features and functionalities available in the IBM mainframe platform that LzLabs did not implement for prior SDM deployments. Thus, in touting the SDM as a product adapted for seamless and immediate integration with IBM-compatible applications, LzLabs misrepresents the potential costs and delays involved in each deployment of the SDM for a new customer with differing functionality requirements.

166. Apart from its lower price, the SDM's suitability for IBM-compatible applications is the SDM's primary selling point for customers. Thus, it is a material consideration for any customer in the market for a mainframe.

167. LzLabs made these false or misleading statements about material features of the SDM to customers both in the U.S. and abroad in an effort to influence those customers to abandon their use of the IBM mainframe and replace it with the SDM.

168. As a direct and proximate result of LzLabs' false advertisements and promotions, some IBM mainframe customers have been misled, and others are likely to be misled, about the adequacy and efficacy of the SDM as a replacement for IBM's mainframe. The reputation of IBM and its mainframe platform is therefore harmed as a result.

169. Development work on the SDM is performed both in Switzerland and in Texas. LzLabs has installed and conducted sales tests for at least one current IBM customer in Tennessee. Thus, the SDM is a product in interstate commerce.

DEMAND FOR JURY TRIAL

IBM demands a jury trial for all issues deemed to be triable by a jury.

PRAYER FOR RELIEF

WHEREFORE, IBM requests the following relief:

A. An injunction restraining LzLabs, Texas Wormhole, and their officers, agents, employees, affiliates, and all other persons in concert with them from further misappropriating or using IBM's trade secrets, from further engaging in acts in violation of the patent laws, and from engaging in further false advertising practices;

B. Entry of judgment holding LzLabs and Texas Wormhole liable for willful and malicious misappropriation of IBM's trade secrets, patent infringement, and false advertising;

C. An award of damages according to proof, including without limitation IBM's lost profits and amounts by which LzLabs and Texas Wormhole has been unjustly enriched, together with prejudgment and post-judgment interest;

D. An award of exemplary and punitive damages and attorneys' fees in light of the willful and malicious nature of LzLabs' and Texas Wormhole's conduct and misappropriation of the IBM trade secrets; and

E. Any and all additional legal and equitable relief that may be available under law and that the court may deem proper.

Dated: March 21, 2022

Respectfully submitted,

THE DACUS FIRM, P.C.

By: */s/ Deron R. Dacus*

Deron R. Dacus
State Bar No. 00790553
The Dacus Firm, P.C.
821 ESE Loop 323, Suite 430
Tyler, TX 75701
Phone/Fax: (903) 705-1117
ddacus@dacusfirm.com

QUINN EMANUEL URQUHART & SULLIVAN, LLP

David Nelson
Nathan Hamstra

davenelson@quinnemanuel.com
nathanhamstra@quinnemanuel.com
191 N. Wacker Drive, Suite 2700
Chicago, Illinois 60606
(312) 705-7400

Alexander Rudis
alexanderrudis@quinnemanuel.com
51 Madison Avenue, 22nd Floor,
New York, New York 10010
(212) 849-7000

Nina Tallon
ninatallon@quinnemanuel.com
1300 I Street NW, Suite 900
Washington, D.C. 20005
(202) 538-8000

Attorneys for Plaintiff International Business Machines Corporation