UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
13/774,140	02/22/2013	Ilie GARBACEA	075803.000045	3947

20230 7590 02/13/2018
Vorys, Sater, Seymour and Pease LLP
1909 K St., NW
9th Floor
WASHINGTON, DC 20006-1152

EXAMINER	
GIROUX, GEORGE	

ART UNIT	PAPER NUMBER
2182	

NOTIFICATION DATE	DELIVERY MODE
02/13/2018	ELECTRONIC

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

patlaw@vorys.com

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

Ex parte ILIE GARBACEA

Appeal 2017-007787
Application 13/774,140
Technology Center 2100

Before JASON V. MORGAN, JEREMY J. CURCURI, and
NABEEL U. KHAN, *Administrative Patent Judges*.

CURCURI, *Administrative Patent Judge*.

DECISION ON APPEAL

Appellant appeals under 35 U.S.C. § 134(a) from the Examiner's rejection of claims 1–18. Final Act. 1. We have jurisdiction under 35 U.S.C. § 6(b).

Claims 1–18 are rejected under 35 U.S.C. § 103(a) as obvious over Abdallah (US 2010/0161948 A1; Jun. 24, 2010) and Zhou (Xiangrong Zhou and Peter Petrov, *Cross-layer customization for rapid and low-cost task preemption in multi-tasked embedded systems*. ACM Trans. Embed. Comput. Syst. 8, 2, Article 14 (January 2009), 28 pages.). Final Act. 3–13.

We affirm.

STATEMENT OF THE CASE

Appellant's invention relates to microprocessors. Spec. ¶ 1. Claim 1 is illustrative and reproduced below:

1. A method of sharing a register pool among a plurality of microprocessor threads using deferred register storage, comprising:

allocating a first set of registers in the register pool to a first thread, wherein the first thread executes a first instruction using the first set of registers in the register pool;

mapping the first set of registers to the first thread;

descheduling the first thread without saving values stored in the first set of registers;

scheduling a second thread to execute a second instruction using registers allocated in the register pool; and

rescheduling the first thread to reuse the first set of registers based on the mapping.

ANALYSIS

The Examiner finds Abdallah and Zhou teach all limitations of claim 1. Final Act. 3–5. The Examiner finds Abdallah teaches all limitations of claim 1 except for "rescheduling the first thread *to reuse the first set of registers based on the mapping*" (claim 1 (emphasis added)), which the Examiner finds is taught by Zhou. Final Act. 4 (citing Zhou 14:17–14:18, section 5).

The Examiner reasons

> It would have been obvious to one of ordinary skill in the art, at the time the invention was made, to reuse the same set of registers when rescheduling a thread, as taught by Zhou, in the system that stores separate register sets for each thread to prevent saving/restoring threads on context switch, taught by Abdallah.
>
> Because both systems disclose providing different mappings/areas for registers for each thread, when space is available, it would have been obvious to one of ordinary skill in the art to reuse the same set of registers when rescheduling a thread, as taught by Zhou, in the system that stores separate register sets for each thread to prevent saving/restoring threads on context switch, taught by Abdallah, to achieve the predictable result of allowing fast context switching when the registers a thread was using are still available, to avoid saving and restoring from memory.

Final Act. 4–5.

Appellant presents the following principal arguments:

> [i] Abdallah does not teach that a first set of registers is mapped to a first thread such that the first thread is descheduled without saving values stored in the first set of registers, and the first thread is rescheduled to reuse the first set of registers based on the mapping. Neither paragraphs [0055]–[0057], [0063]–[0070], or Fig. 4 disclose such a feature. Instead, Abdallah teaches that the architecture state may be saved and restored gradually as the new context is replacing the old context. In other words, the individual registers are swapped/read in and out by the hardware upon the use of that particular register in the new or old context. See paragraph [0031].

App. Br. 7; *see also* Reply Br. 2–3 ("the Examiner's Answer alleges, without any evidentiary support or reasoning, that the mere mention in Abdallah of saving registers to memory only when the space is needed by a new thread implies that when there is available register space a thread's registers remain and can be used again upon rescheduling the thread").

> [ii] The rejection relies on pages 14:17 and 14:18 as disclosing reusing a mapped first set of registers when rescheduling a thread (final rejection at 4). However, Zhou does not disclose descheduling the first thread without saving values stored in the first set of registers. To the contrary, as cited above Zhou explicitly teaches the opposite, that upon thread preemption the entire content of the register file is saved.

App. Br. 8; *see also* Reply Br. 3–4.

In response, the Examiner explains

> While the fact that Abdallah specifically mentions saving registers to memory only when the space is needed by a new thread, and otherwise allocating free register space to the new thread, implies that when there is available register space a thread's registers remain (they have not been needed by a new thread, and thus not moved to memory) and can be used again upon resheduling the thread, it is not explicitly taught by the reference, and thus the examiner has relied upon the combination with Zhou to provide the teaching of rescheduling a thread to reuse the first set of registers based on the mapping.

Ans. 14.

The Examiner also explains

> Zhou teaches that using a register-mapped context switch (RMCS) technique can eliminate cycles required to save registers to memory upon context switches by keeping the live set of registers for the task being preempted in the register file by quickly remapping the address space of the register file where the registers reside, in effect hiding these registers (in the register file) until they are needed again while the live registers of the preempting task, which are still in the register file, are mapped back for that task (i.e., upon one task replacing another, only the mapping of registers in the register file needs to change, rather than saving/restoring registers to/from memory) (see Zhou: pgs. 14:17–14:18, section 5; etc.).

Ans. 15.

We review the appealed rejections for error based upon the issues identified by Appellants, and in light of the arguments and evidence produced thereon. *Ex parte Frye*, 94 USPQ2d 1072, 1075 (BPAI 2010) (precedential).

We do not see any error in the Examiner's findings. Nor do we see any error in the Examiner's conclusion of obviousness.

Abdallah discloses, for example,

> One other scheme that allows for establishing a virtual register file is the aliasing of the registers of different contexts/threads that need a smaller number of registers than that provided by the instruction set architecture. For example, a small function call or a worker thread might only need to use a subset of all the registers e.g., 8 registers (register 0 to 7) out of the 32 registers available by the instruction set. If this is the case, then the hardware/compiler will alias this 8 logical register batch on top of another free physical register batch, the register batch that the thread gets mapped to does not need to be with the same register numbers; i.e., registers 0-7 can be mapped onto physical registers 8-15 or 16-23 or 24-31 that are available in another context physical state storage.

Abdallah ¶ 55.

Thus, Abdallah teaches (claim 1)

> allocating a first set of registers in the register pool to a first thread, wherein the first thread executes a first instruction using the first set of registers in the register pool;
>
> mapping the first set of registers to the first thread;
>
> descheduling the first thread without saving values stored in the first set of registers;
>
> scheduling a second thread to execute a second instruction using registers allocated in the register pool; and
>
> rescheduling the first thread

because Abdallah's first thread uses registers 0 to 7, and Abdallah's second thread uses registers 8 to 15 by mapping (aliasing) registers 0 to 7 to registers 8 to 15. *See* Abdallah ¶ 55.

Zhou discloses, for example,

> the RMCS approach keeps the live set [of registers] in the register file by quickly remapping the address space of the register file where the live registers reside. In effect the live register[s] of the preempted task are being hidden, while the live registers of the preempting task are mapped back (mounted) to the register file space which is ISA-visible.

Zhou 14:18.

Thus, Zhou teaches (claim 1 (emphasis added)) "rescheduling the first thread *to reuse the first set of registers based on the mapping*" because Zhou's preempting task reuses its live registers, which were kept in the register file, when they are mounted to the register file space. *See* Zhou 14:18.

When Abdallah is modified in light of Zhou's teachings, Abdallah's rescheduling of the first thread reuses registers 0 to 7. *See* Abdallah ¶ 55, Zhou 14:18.

Appellant's argument (i) is unavailing because Abdallah at ¶ 55 discloses Abdallah's first thread uses registers 0 to 7, and Abdallah's second thread uses registers 8 to 15 by mapping (aliasing) registers 0 to 7 to registers 8 to 15. At this instant, the first thread's registers remain in the register file. With regard to keeping the first thread's registers available in the register file for when the first thread is rescheduled, this is taught by Zhou at 14:18.

Appellant's argument (ii) is unavailing because Abdallah at ¶ 55 teaches descheduling the first thread without saving values. Further, Zhou at

14:18 teaches hiding the live registers of the preempted task. Thus, these live registers maintain their contents and are reused when the preempted task is rescheduled.

The Examiner articulated a reason to combine the teachings of the reference which is rational on its face and supported by evidence drawn from the record. *See* Final Act. 5 ("achieve the predictable result of allowing fast context switching when the registers a thread was using are still available, to avoid saving and restoring from memory."); *see also* Zhou 14:18.

We, therefore, sustain the Examiner's rejection of claim 1. We also sustain the Examiner's rejection of claims 2–18, which are not separately argued with particularity.

ORDER

The Examiner's decision rejecting claims 1–18 is affirmed.

No time period for taking any subsequent action in connection with this appeal may be extended under 37 C.F.R. § 1.136(a)(1).

AFFIRMED