Zeszyty Naukowe
Wyższej Szkoły Ekonomii i Informatyki
Nr 20/2024, s. 119-132
DOI: XXXXXX

ISSN 1734-5391
e-ISSN 2957-2223

Piotr Górski
mgr inż.
Wyższa Szkoła Ekonomii i Informatyki
w Krakowie (WSEI)
email: pgorski@wsei.edu.pl
ORCID: 0009-0003-6736-9750

Zbigniew Handzel
dr inż., prof. WSEI
Wyższa Szkoła Ekonomii i Informatyki
w Krakowie (WSEI)
email: zhandzel@wsei.edu.pl
ORCID: 0000-0003-1470-6592

ITERACJE ARCHITEKTURY X86_64

ITERATIONS OF THE X86_64 ARCHITECTURE

Słowa kluczowe: *architektura x86, architektura x86_64, Intel, AMD*
Key words: *x86 archietecture, x86_64 architecture, Intel, AMD*

JEL Classification: A2, C8, I2, L2, P4

Streszczenie

Niniejszy artykuł dotyczy rozważań nad istotą oraz sensem wdrożenia iteracji architektury x86_64. Autorzy podkreślają, że omawiane w artykule iteracje architektury 64 bitowej nie są niczym nowym, jednak stale rosnące zapotrzebowanie na wydajność obliczeniową skłoniło korporacje do użycia niszowych do tej pory rozwiązań. W efekcie tego, w coraz większej liczbie znanych i uznanych projektów, takich jak np.: Red Hat Linux Enterprise, openSUSE czy Ubuntu rozważa się wprowadzenie źródeł skompilowanych w architekturze x86_64_v2 oraz x86_64_v3. W niniejszym artykule przedstawiono zarówno historię architektury x86_64 oraz wprowadzenie

do iteracji architektury x86_64, by lepiej zaznajomić czytelnika z omawianą problematyką, jak również omówiono wybrane przykłady wykorzystania architektury x86_64_v2 oraz x86_64_v3, a także przedstawiono wyniki badań.

Abstract
This article focuses on the considerations regarding the essence and purpose of implementing iterations of the x86_64 architecture. The authors emphasize that the iterations of the 64-bit architecture discussed in the article are not new; however, the ever-increasing demand for computational performance has prompted corporations to adopt solutions that were previously considered niche. As a result, an increasing number of well-known and established projects, such as Red Hat Linux Enterprise, openSUSE, and Ubuntu, are considering the introduction of sources compiled in the x86_64_v2 and x86_64_v3 architectures. This article presents both the history of the x86_64 architecture and an introduction to its iterations to better acquaint the reader with the subject matter. Additionally, it discusses selected examples of the use of the x86_64_v2 and x86_64_v3 architectures and presents research findings.

WPROWADZENIE

W ostatnich miesiącach w coraz większej liczbie znanych i uznanych projektów, takich jak Red Hat Linux Enterprise, openSUSE czy Ubuntu rozważa się wprowadzenie źródeł skompilowanych w architekturze x86_64_v2 oraz x86_64_v3. Oczywiście wspomniane iteracje architektury 64 bitowej nie są niczym nowym i liczą sobie już ponad 10 lat, jednak rosnące zapotrzebowanie na wydajność skłoniło korporacje do użycia niszowych do tej pory rozwiązań.

W niniejszym artykule autorzy postanowili przedstawić rozważania nad istotą oraz sensem wdrożenia iteracji architektury x86_64. Na początek warto zaznaczyć, iż jeden z autorów niniejszej publikacji od wielu lat jest developerem, zaangażowanym w rozwój systemów Linuxowych, a obecnie bierze czynny udział w rozwijaniu systemu operacyjnego CachyOS, który jako jeden z pierwszych systemów operacyjnych udostępnia źródła skompilowane w architekturze x86_64_v3. Dzięki temu autorzy w swoich rozważaniach mają bezpośredni dostęp do jednego z ważniejszych projektów i mogą dokonać starannej analizy zalet oraz wad wynikających ze

stosowania tych rozwiązań. Warto zauważyć, że jeszcze do niedawna mało popularne rozwiązanie zdobywa coraz większą liczbę zwolenników. Należy zatem zadać pytanie o jego sens i należy się zastanowić, czy ma to realny wpływ na działanie systemu? Zdaniem autorów niniejszej publikacji odpowiedź na tak zadane pytanie jest twierdząca. Jak już wspomniano, jeden z autorów jest zaangażowany w rozwój systemu operacyjnego CachyOS, gdzie od dwóch lat korzysta on z takich rozwiązań, a przy tym stale ma styczność z użytkownikami, otrzymując od nich opinie oraz uwagi. Dzięki temu autorzy niniejszej publikacji mogą lepiej poznać istotę i sens stosowania omawianych rozwiązań. Korzystanie z tych architektur ma oczywiście swoje wady i zalety, jednakże te ostatnie zdecydowanie przeważają nad tymi pierwszymi. Możliwość zbudowania całego systemu operacyjnego pod posiadaną architekturę uczyni środowisko pracy bardziej wydajnym. Oczywiście, jak wspomniano, istnieją też wady, takie jak większy pobór energii, czy potencjalne błędy – wszakże istnieją projekty w których nie ma póki co możliwości zastosowania takich optymalizacji lub też mogą one implikować potencjalne regresje. Rosnące zainteresowanie zarówno ze strony korporacji jak i użytkowników pokazuje jednak, że najprawdopodobniej niszowe dziś rozwiązanie może stać się obowiązującym standardem na długie lata. Architektura x86_64 może alokować w teorii aż 16 EB pamięci operacyjnej co znacznie przekracza dzisiejsze możliwości zarówno sprzętowe jak i systemowe. Oznacza to, że najprawdopodobniej jeszcze długo nie ukaże się architektura 128 bitowa przeznaczona do powszechnego użytku. W takim wypadku opisywane rozwiązania stanowią analogię do rozwoju architektury i386, która z czasem wyewoluowała w architektury i486/i586/i686.

HISTORIA ARCHITEKTURY X86_64

Architektura 64 bitowa (określana jako x86_64 lub też AMD64) jest rozwiązaniem liczącym sobie już dwie dekady. Opracowana i zapoczątkowana przez firmę AMD (stąd też wzięło się jedno z określeń – amd64) [1] jest rozwijana również przez firmę Intel [2] pod symbolem EM64T. Kompatybilność pomiędzy tymi rozwiązaniami nie jest pełna, dodatkowo to AMD64 było zdecydowanie lepiej dopracowane. To właśnie implementacja zapoczątkowana przez AMD stała się obowiązującym standardem i podwaliną do kolejnych iteracji architektury x86_64. Jakkolwiek architektura 64 bitowa jest dzisiaj powszechnie używana i zdecydowanie wy-

parła architekturę 32 bitową – czego przykładem może być wycofywanie się różnych projektów ze wsparcia [3], tak powszechna świadomość na temat iteracji tejże architektury nie jest powszechna. Świadczyć o tym może choćby fakt, iż oficjalne wsparcie dla kompilatora pojawiło się dopiero pod koniec 2020 roku [4] – pomimo, że pierwsza iteracja tej architektury, określana jako x86_64_v2 pojawiła się już w 2009 roku [5] w procesorach Intel Nehalem oraz AMD Jaguar. Obowiązującym dzisiaj standardem jest druga iteracja – x86_64_v3 wprowadzona w latach 2014/2015 w procesorach Intel Haswell oraz AMD Excavator [5]. Jest to też ostatni przypadek, w którym kolejne rozwinięcie architektury 64 bitowej zostało wydane przez firmy Intel oraz AMD w podobnym czasie. Kolejna, ostatnia już iteracja – określona wersją x86_64_v4 została zapoczątkowana przez Intela już w procesorach Intel SkylakeX w 2017 roku – i było to pierwsze podejście do tej architektury [5]. Kolejne nastąpiło w 11 generacji procesorów Core i5/i7/i9 [6] lecz bardzo szybko, bo już w 12 generacji Intel wycofał się ze stosowania flagi AVX512 – powracając zarazem do architektury x86_64_v3), stosując w zamian tego nowe rozwiązanie, bazujące na dwóch rodzajach rdzeni – E oraz P [6]. AMD natomiast wprowadziło architekturę x86_64_v4 dopiero pod koniec 2022 roku w procesorach z rodziny ZEN4 [7]. Intel swoją decyzję argumentował faktem, że flaga AVX512 nie jest rozwiązaniem korzystnym dla użytkowników komputerów osobistych, jest za to odpowiednim dla serwerów [8]. Bardzo negatywnie na temat architektury x86_64_v4 wypowiadał się twórca jądra Linux, Linus Torvalds: *„AVX512 ma prawdziwe wady. Wolałbym, żeby budżet na tranzystory został wykorzystany na inne rzeczy, które są o wiele bardziej istotne. Nawet jeśli nadal jest to matematyka FP (w GPU, a nie AVX512). Albo po prostu dać mi więcej rdzeni (z dobrą wydajnością jednowątkową, ale bez śmieciowych rozwiązań, takich jak AVX512), tak jak zrobiło to AMD"* [9].

WPROWADZENIE DO ITERACJI ARCHITEKTURY X86_64

Z powyższych względów w niniejszym artykule warto skupić się na architekturach x86_64 w wersjach: v2 v3 – a w szczególności na tej ostatniej, gdyż jest ona nie tylko najnowsza, ale również jest wspólną generacją dla większości procesorów Intela oraz AMD z lat 2015-2023. Rozważania nad koniecznością wdrożenia iteracji architektury 64 bitowej podjęto między innymi w Red Hat [10]. Taki wybór oznaczałoby użycie następujących flag procesora: CMPXCHG16B, LAHF-SAHF, POPCNT, SSE3, SSE4.1, SSE4.2, SSSE3 – czyli flag udostępnionych wraz z Intel Nehalem. To z kolei spo-

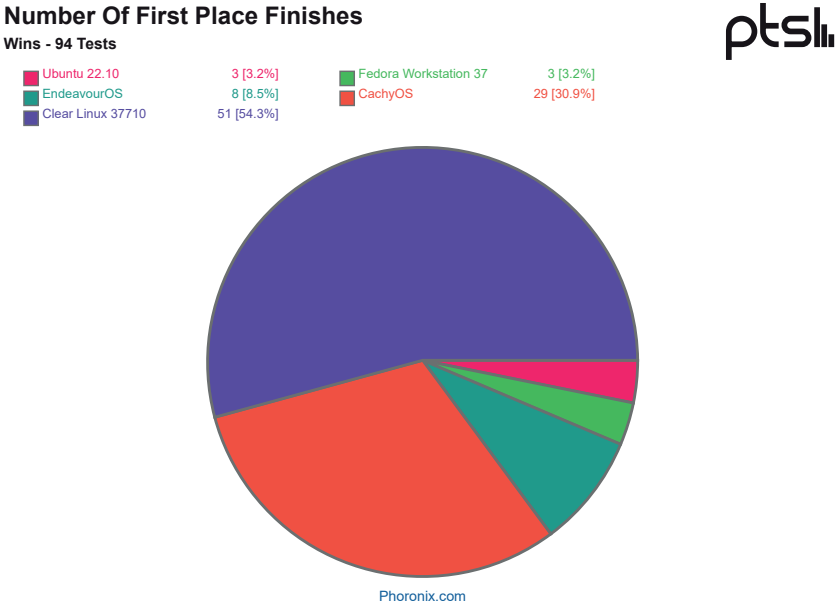
wodowałoby, że RHEL9 zachowałby kompatybilność z ponad dziesięcioletnimi komputerami – co i tak należy uznać za bardzo liberalne podejście, gdyż iteracja x86_64_v3 wymusiłaby odcięcie znacznej liczby komputerów od możliwości zaktualizowania systemu do najnowszej wersji [10]. Developerzy Red Hata słusznie zauważyli, że zastosowanie większej ilości flag znacząco podniesie wydajność systemu operacyjnego już na samym początku [10]. RHEL10 natomiast poszedł o krok dalej i wprowadził zgodność z architekturą x86_64_v3 [11]. Na podobny krok zdecydowała się firma Canonical w nadchodzącym Ubuntu 24.04 LTS [12]. Oczywiście, zarówno Red Hat jak i Canonical nie zamierzają porzucać bazowej architektury x86_64, natomiast zamierzają dodać jedynie wsparcie architektury x86_64_v3. Bardziej zaawansowane prace w tym kierunku prowadzą developerzy systemu openSUSE i zdecydowali się oni wprowadzić implementację architektury x86_64_v3 wcześniej niż Red Hat, czy Canonical [12]. W tym wypadku pozostawiono jednak dobrowolną decyzję użytkownikowi, który samodzielnie wybiera, czy chce z takiej funkcjonalności skorzystać czy też nie [13]. Jeśli użytkownik nie chciał lub też nie mógł korzystać z takiej formy optymalizacji, wystarczyło odinstalować pakiet „patterns-glibc-hwcaps-x86_64_v3", który nie był ponownie automatycznie aktualizowany [14]. Podobne zamiary mieli także developerzy systemu operacyjnego Arch Linux [14], jednakże ostatecznie nie zostało to oficjalnie wdrożone. Istnieje natomiast projekt, w którym podjęto pracę nad takim rozwiązaniem i udostępniono pakiety dla Arch Linux budowane zarówno w architekturze x86_64_v2 jak i x86_64_v3 [15].

PRZYKŁADY UŻYCIA ARCHITEKTURY X86_64_V2 ORAZ X86_64_V3

Jedynym systemem operacyjnym, który do tej pory wprowadził pełną obsługę pakietów zbudowanych w architekturze x86_64_v3 jest bazujący na Arch Linux – CachyOS [16]. W tej dystrybucji planowana jest także obsługa architektury x86_64_v4 [17]. Wspomniany CachyOS został zestawiony z takimi systemami, jak Clear Linux (zoptymalizowana pod benchmarki dystrybucja tworzona przez Intela), EndeavourOS (system również bazujący na Arch Linux), Fedora (dystrybucja sponsorowana przez Red Hat) oraz Ubuntu (tworzone przez Canocnical). W teście wydajnościowym przeprowadzonym przez portal Phoronix system CachyOS zajął drugie miejsce za systemem Clear Linux [18]. Wyniki tego testu

pokazały, iż używanie architektury x86_64_v3 potrafi znacząco podnieść wydajność systemu operacyjnego. Może mieć to wielorakie zastosowanie – od tworzenia bardziej wydajnych aplikacji i systemów operacyjnych dla klienta końcowego, po używanie takiego środowiska pracy we własnym zakresie. Zwiększona wydajność jest pożądanym efektem końcowym zarówno do pracy jak i do rozrywki. Można zatem powiedzieć, iż użycie x86_64_v3 zarówno w biznesie jak i do celów prywatnych jest bardzo korzystnym rozwiązaniem.

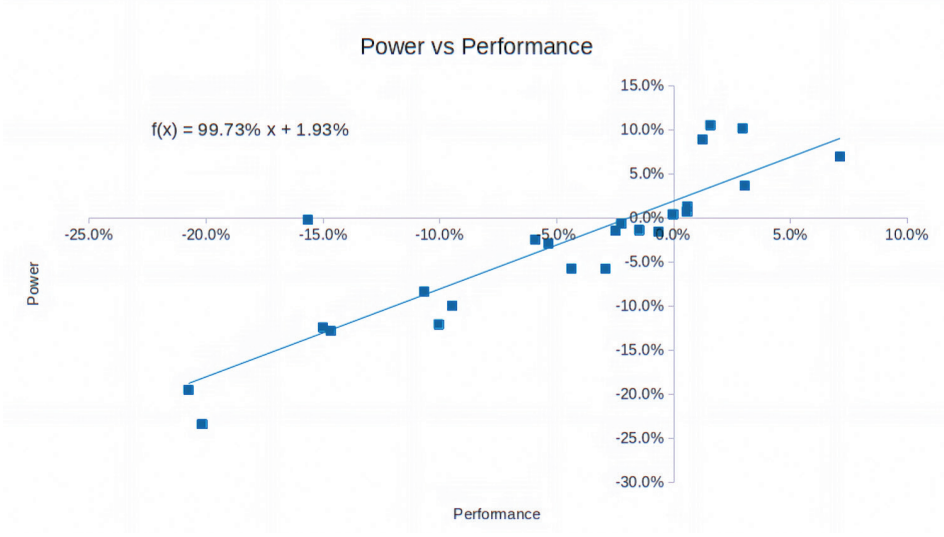
Rysunek 1. Wyniki testów wydajnościowych przedstawione na stronie Phoenix



Źródło: https://www.phoronix.com/review/cachyos-linux-perf/5.

W kolejnym teście, który został przeprowadzony przez użytkownika portalu Github, zestawiono standardowe pakiety z Arch Linux (skompilowane w generycznej architekturze x86_64) z pakietami CachyOS (zoptymalizowanymi pod architekturę x86_64_v3) [19]. W swoim teście Peter O'Connor wykazał, iż korzystanie z optymalizacji przeważnie jest korzystne, natomiast wadą stosowania architektury x86_64_v3, jest znacznie większy pobór mocy sprzętu. Finalnie jednak otrzymuje się lepszą ogólną wydajność [20].

Rysunek 2. Porównanie poboru mocy oraz wydajności dla analizowanych architektur



Źródło:https://raw.githubusercontent.com/sunnyflunk/sunnyflunk.github.io/main/_posts/img/20230115.png.

Warto podkreślić, że nie wszystkie pakiety można optymalizować przy użyciu architektury x86_64_v3. Przykładem tutaj może być projekt ALHP [21]. W przytoczonym przykładzie niektóre pakiety nie budują się, a inne działają wadliwie, w efekcie czego zostały dodane do blacklisty.

Aby zapewnić prawidłowe działanie systemu operacyjnego skompilowanego w architekturze innej niż x86_64, konieczne jest zastosowanie kilku zmian, między innymi w menedżerze pakietów. Za przykład posłuży tutaj dystrybucja CachyOS, która w ocenie autorów niniejszego artykułu wprowadziła najlepsze i najbezpieczniejsze dla użytkownika końcowego rozwiązania. Kluczowym czynnikiem jest wprowadzenie mechanizmu autodetekcji architektur. We wspomnianej dystrybucji CachyOS, wprowadzono odpowiednie rozwiązanie [22]. Autodetekcja wykrywa flagi procesora na zainstalowanym systemie operacyjnym [22] oraz na tej podstawie dokonuje detekcji odpowiedniej architektury [23]. Jak można zauważyć, architektury te cechują się wsteczną kompatybilnością – zatem architektury x86_64_v2 można używać mając sprzęt wspierający x86_64_v3. Mając natomiast sprzęt kompatybilny z architekturą x86_64_v4, można używać

także x86_64_v2 oraz x86_64_v3. Można zatem postawić pytanie: Co daje taka autodetekcja? Otóż, uniemożliwia ona zainstalowanie niewspieranej architektury – jeśli dysponujemy sprzętem obsługującym maksymalnie x86_64_v3, nie będzie możliwości zainstalowania x86_64_v4.

Kolejnym ważnym krokiem podjętym przez developerów CachyOS było umożliwienie użytkownikom budowania pakietów w architekturach innych niż x86_64. Do kodu źródłowego została dodana nowa opcja i w jej efekcie w ustawieniach menedżera pakietów pojawił się adekwatny wpis [24]. Domyślnie przyjmuje ona wartość generyczną i użytkownik wedle potrzeb może ją zmienić. Najpierw jednak należy sprawdzić w systemie operacyjnym jaką architekturę obsługuje posiadany sprzęt komputerowy.

Fragment kodu 1. Sposób sprawdzenia posiadanej architektury x86_64 na systemie Linux

```
lucjan at cachyos ~ 15:10:39
❯ /lib/ld-linux-x86-64.so.2 --help | grep supported
    STDIN
    Subdirectories of glibc-hwcaps directories, in priority order:
      x86-64-v4
      x86-64-v3 (supported, searched)
      x86-64-v2 (supported, searched)
lucjan at cachyos ~ 15:10:48
❯
```

Źródło: opracowanie własne.

Mając już informację na temat posiadanej architektury, można dokonać modyfikacji ustawień menedżera pakietów.

Fragment kodu 2. Modyfikacja menedżera pakietów

```
lucjan at cachyos ~ 15:14:34
❯ cat /etc/makepkg.conf | grep PACKAGECARCH
    STDIN
CHOST="x86_64-pc-linux-gnu"
#-- That will be applied only at the package() stage.
PACKAGECARCH="x86_64_v3"

#-- Compiler and Linker Flags
lucjan at cachyos ~ 15:14:56
❯
```

Źródło: opracowanie *własne.*

Krótki eksperyment pozwoli ocenić, czy dane rozwiązanie działa prawidłowo. Eksperyment zostanie podzielony na dwa etapy. W pierwszym zostanie sprawdzona detekcja posiadanej architektury (x86_64_v3) natomiast w drugim niewspieranej (x86_64_v4).

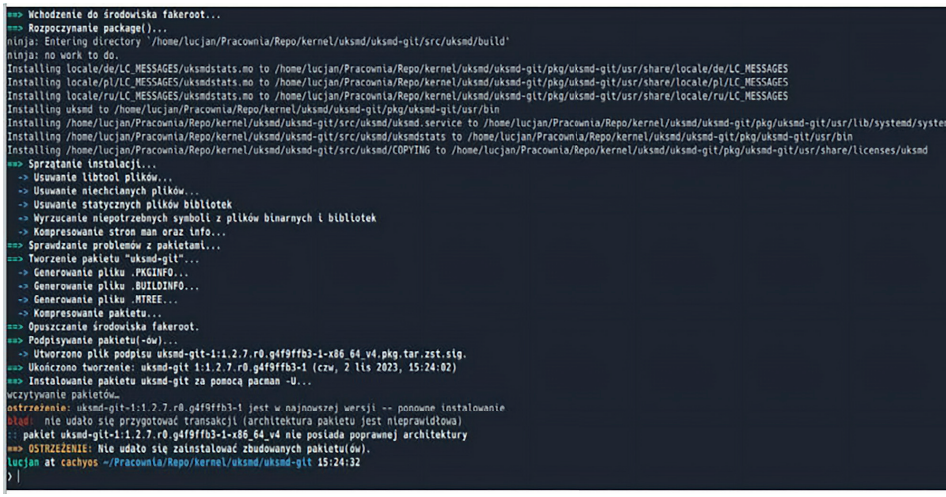
Pierwszy etap eksperymentu zakończył się pełnym powodzeniem i paczka została zainstalowana w systemie operacyjnym.

Fragment kodu 3. Udana próba zainstalowania pakietu zbudowanego w architekturze x86_64_v3

Źródło: opracowanie własne.

Próba zainstalowania paczki zbudowanej x86_64_v4 zakończyła się natomiast niepowodzeniem.

Fragment kodu 4. Nieudana próba zainstalowania pakietu zbudowanego w architekturze x86_64_v4



Źródło: opracowanie własne.

W tym jednak wypadku niepowodzenie drugiej fazy eksperymentu oznacza, że detekcja architektury została zaimplementowana prawidłowo i dlatego zablokowano możliwość zainstalowania programu zbudowanego w niewspieranej architekturze.

PODSUMOWANIE

Jak pokazano w niniejszej publikacji, używanie x86_64_v2 oraz x86_64_v3 w miejsce generycznego x86_64 niesie ze sobą zarówno korzyści jak i potencjalne regresje. Niemniej jednak, testy przeprowadzone przez portal Phoronix wykazały, że system skompilowany w architekturze x86_64_v3 w przeważającej części testów działa zdecydowanie wydajniej niż taki, który został skompilowany x86_64. Iteracje te zostały dostrzeżone po latach przez upstream oraz takie korporacje jak RedHat (IBM), czy SUSE (Micro Focus). Wykazały one zainteresowanie optymalizacją swojego oprogramowania, tworzonego w tych architekturach. Problemy z kompatybilnością wsteczną nie będą tutaj przeszkodą, gdyż ciężko jest nie zgodzić się ze stwierdzeniem, że większość użytkowników zarówno komercyjnych

jak i prywatnych posiada sprzęt obsługujący zarówno x86_64_v3 (lata 2014/2015), a tym bardziej x86_64_v2 (lata 2009/2010).

Autorzy niniejszego artykułu uważają, że korzyści płynące z tego rozwiązania odczują wszyscy, ponieważ oprogramowanie zoptymalizowane pod konkretną architekturę będzie działało zauważalnie lepiej, o czym wspomniano wcześniej, a dodatkowo zostało to wykazane przez testy porównawcze. Architektura x86_64_v4 przez długi czas pozostanie natomiast raczej ciekawostką niż realnym rozwiązaniem stosowanym w biznesie. Procesory Intela od 12 generacji ponownie nie wspierają tej architektury, a AMD zaimplementowało to dopiero pod koniec 2022 roku. Rozwiązanie to cechuje się bardzo ograniczoną ilością wspieranego sprzętu, brakiem kompatybilności ze starszymi architekturami (w artykule wykazano, że architektury x86_64_v4 nie da się zastosować na procesorach wspierających x86_64_v3 oraz starszych) oraz krytycznym podejściem osób cenionych w branży IT, takich jak chociażby Linus Torvalds. Wiele wskazuje więc na to, iż to x86_64_v3 zostanie spopularyzowane i na długie lata pozostanie powszechnie używaną architekturą. Dziwić natomiast może fakt, że dopiero od 3 lat istnieje możliwość pełnego i bezproblemowego korzystania z tych rozwiązań, choć pierwsze implementacje tego powstały już ponad 10 lat temu. Z drugiej zaś strony, nawet po wprowadzeniu architektury x86_64 przez długie lata wielu użytkowników dalej używało architektury 32 bitowej (i386/i486/i586/i686) i dopiero po porzuceniu tego rozwiązania przez upstream, zostali oni niejako zmuszeni do migracji na nowe rozwiązania. Podobnie będzie i w tym przypadku – postęp technologiczny wymaga długich lat testów, by końcowe rozwiązanie działało stabilnie oraz prawidłowo. Po dwóch latach używania wyżej wymienionych rozwiązań autorzy publikacji uważają, iż w przypadku stosowania iteracji x86_64_v3, jej zalety zdecydowanie przewyższają wady. Zainteresowania korporacji takich jak Red Hat jasno pokazują, że to nieunikniona przyszłość i już niedługo takie rozwiązania staną się najprawdopodobniej obowiązującym standardem.

Biorąc pod uwagę przeprowadzane testy można jednoznacznie stwierdzić, że stosowanie architektur x86_64_v2/v3 jest dobrym posunięciem. Zalety tego rozwiązania są zdecydowanie większe niż wady, wynikające z braku wsparcia dla tego rozwiązania w przypadku części oprogramowania. Warto jednak pamiętać o kompatybilności wstecznej, która umożliwia korzystanie z oprogramowania budowanego w oparciu o x86_64 w środowisku zbudowanym w standardzie x86_64_v3. Z pewnością popularyzacja tego rozwiązania sprawi, że twórcy oprogramowania w swoich projektach

z czasem usuną związane z tym problemy, więc za kilka lat najprawdopodobniej o tych niedogodnościach nikt nie będzie już pamiętał. Kolejnym atutem tego rozwiązania jest fakt, że działa ono zarówno na procesorach AMD oraz Intela, dokładnie tak samo jak ma to miejsce w przypadku generycznej architektury x86_64. Problemy z implementacją iteracji x86_64_v4 sprawią, że to właśnie v3 na długie lata zastąpi klasyczną architekturę 64 bitową. Jak już zostało wspomniane, architektura ta teoretycznie może zaalokować aż 17 miliardów GB pamięci operacyjnej, co najpewniej wyklucza konieczność wprowadzenia architektury 128 bitowej do powszechnego użytku. Wsparcie korporacji takich jak Red Hat czy Canonical, które są liderami w dostarczaniu rozwiązań serwerowych spowoduje, że architektura x86_64_v3 szybko trafi do głównego nurtu branży IT. Nieograniczony popyt na coraz większą wydajność środowiska pracy wymusza wprowadzanie coraz bardziej innowacyjnych rozwiązań, a przykład CachyOS pokazuje, że zastosowanie tej architektury jako głównej znacząco podnosi wyniki uzyskiwane w testach wydajnościowych, nie wpływając zarazem negatywnie na stabilne działanie.

BIBLIOGRAFIA

1. https://www.pcmag.com/encyclopedia/term/amd64 (stan na dzień: 02.11.2023).
2. https://www.intel.com/content/www/us/en/support/articles/000005898/processors.html (stan na dzień: 02.11.2023).
3. https://archlinux.org/news/phasing-out-i686-support/ (stan na dzień: 02.11.2023).
4. https://www.phoronix.com/news/GCC-11-x86-64-Feature-Levels (stan na dzień: 02.11.2023).
5. https://gitlab.com/x86-psABIs/x86-64-ABI/-/blob/master/x86-64-ABI/low-level-sys-info.tex (stan na dzień: 02.11.2023).
6. https://www.makeuseof.com/what-is-avx-512-why-intel-killing-it/ (stan na dzień: 02.11.2023).
7. https://gitlab.com/x86-psABIs/x86-64-ABI/-/blob/master/x86-64-ABI/low-level-sys-info.tex (stan na dzień: 02.11.2023).
8. https://www.makeuseof.com/what-is-avx-512-why-intel-killing-it/ (stan na dzień: 02.11.2023).
9. https://www.phoronix.com/news/Linus-Torvalds-On-AVX-512 (stan na dzień: 02.11.2023).

10. https://developers.redhat.com/blog/2021/01/05/building-red-hat-enterprise-linux-9-for-the-x86-64-v2- microarchitecture-level#architectural considerations for rhel 9 (stan na dzień: 02.11.2023).
11. https://www.phoronix.com/news/RHEL-9-x86-64-v2-Plans (stan na dzień: 02.11.2023).
12. https://www.phoronix.com/news/RedHat-RHEL10-x86-64-v-3-Explore (stan na dzień: 15.01.2024).
13. https://www.phoronix.com/review/ubuntu-x86-64-v3-benchmark (stan na dzień: 15.01.2024).
14. https://news.opensuse.org/2023/03/02/tw-gains-optional-optimizations/ (stan na dzień: 02.11.2023).
15. https://www.phoronix.com/news/openSUSE-TW-x86-64-v3-RPM (stan na dzień: 02.11.2023).
16. https://www.phoronix.com/news/Arch-Linux-x86-64-v3-Port-RFC (stan na dzień: 02.11.2023).
17. https://somegit.dev/ALHP/ALHP.GO (stan na dzień: 02.11.2023).
18. https://wiki.cachyos.org/cachyos repositories/what is the cachyos repo/ (stan na dzień: 02.11.2023)
19. https://www.phoronix.com/review/cachyos-linux-perf (stan na dzień 02.11.2023).
20. https://www.phoronix.com/review/cachyos-linux-perf/5 (stan na dzień: 02.11.2023).
21. https://github.com/sunnyflunk (stan na dzień: 02.11.2023).
22. https://sunnyflunk.github.io/2023/01/15/x86-64-v3-Mixed-Bag-of--Performance.html (stan na dzień: 02.11.2023).
23. https://alhp.dev/packages.html (stan na dzień: 02.11.2023).
24. https://github.com/CachyOS/pacman/commit/076bd5590f-2f4a9684b07f2d4d40513dbd0ed4a7 (stan na dzień: 02.11.2023)