Cuprins

4. Arhitecturi RISC

- Introducere
- Cauze ale complexității arhitecturale
- Avantaje ale arhitecturilor RISC
- Ferestre suprapuse de registre
- Caracteristici ale arhitecturilor RISC
- Comparație între arhitecturile RISC și CISC
- Arhitectura ARM

Introducere (1)

- Arhitecturile de calculatoare au evoluat progresiv spre o complexitate mai ridicată
  - Număr mare de instrucțiuni
  - Număr mare de moduri de adresare
  - Putere de calcul ridicată a instrucțiunilor individuale
  - Registre specializate
- Calculator cu set complex de instrucțiuni (CISC – *Complex Instruction Set Computer*)

Introducere (2)

- Efectele unei instrucțiuni complexe trebuie evaluate înainte de adăugarea acesteia la setul de instrucțiuni
  - Unele din instrucțiunile procesoarelor CISC sunt utilizate rareori de compilatoare
- Principiul de a nu se adăuga instrucțiuni utilizate rar la setul de instrucțiuni: calculator cu set redus de instrucțiuni (RISC – *Reduced Instruction Set Computer*)

Arhitecturi RISC

- Introducere
- Cauze ale complexității arhitecturale
- Avantaje ale arhitecturilor RISC
- Ferestre suprapuse de registre
- Caracteristici ale arhitecturilor RISC
- Comparație între arhitecturile RISC și CISC
- Arhitectura ARM

Cauze ale complexității arhitecturale

- Facilitarea utilizării limbajelor de nivel înalt
  - S-au prevăzut instrucțiuni mai puternice pentru limbajele de nivel înalt
- Migrarea funcțiilor de la implementarea prin software la implementarea prin hardware
  - Software → firmware → hardware
- Compatibilitatea în sus
  - Un mod de a îmbunătăți un sistem prin adăugarea unor facilități noi, mai complexe

Arhitecturi RISC

- Introducere
- Cauze ale complexității arhitecturale
- Avantaje ale arhitecturilor RISC
- Ferestre suprapuse de registre
- Caracteristici ale arhitecturilor RISC
- Comparație între arhitecturile RISC și CISC
- Arhitectura ARM

Avantaje ale arhitecturilor RISC (1)

- Obiective universal acceptate de către proiectanții de calculatoare:
  - Minimizarea timpului de execuție
  - Minimizarea costului de proiectare
- Metode pentru îndeplinirea obiectivelor:
  - Îmbunătățirea tehnologiei componentelor
  - Minimizarea numărului mediu al ciclurilor de ceas pe instrucțiune
  - Execuția simultană a mai multor instrucțiuni

Avantaje ale arhitecturilor RISC (2)

- Arhitecturile CISC necesită o logică mai complexă
  - Este necesar un microprogram complex
  - O parte importantă a suprafeței circuitului este ocupată de memoria de microprogram
- Suprafața circuitului necesară pentru unitatea de control: CISC $\rightarrow$ 40% .. 60%; RISC $\rightarrow$ 10%
- Suprafața rămasă la o arhitectură RISC poate fi utilizată pentru alte componente

Avantaje ale arhitecturilor RISC (3)

- Viteza de calcul
  - Arhitecturile RISC sunt mai potrivite pentru utilizarea sistemelor *pipeline* de instrucțiuni
- Implementarea în circuite VLSI
  - Unitatea de control a arhitecturilor RISC este implementată prin hardware (cablată)
  - Un număr mare de registre și memoriile *cache* reduc numărul acceselor la memorie

Avantaje ale arhitecturilor RISC (4)

- Timpul de proiectare
  - Timpul necesar pentru testare și depanare este mai redus
  - Posibilitatea erorilor este mai redusă
  - Avantaje: costuri de proiectare mai reduse; fiabilitate de proiectare mai ridicată
- Facilitarea utilizării limbajelor de nivel înalt
  - Instrucțiuni apropiate semantic de caracteristicile limbajelor de nivel înalt

Arhitecturi RISC

- Introducere
- Cauze ale complexității arhitecturale
- Avantaje ale arhitecturilor RISC
- Ferestre suprapuse de registre
- Caracteristici ale arhitecturilor RISC
- Comparație între arhitecturile RISC și CISC
- Arhitectura ARM

Ferestre suprapuse de registre (1)

- Instrucțiunile de apel și revenire consumă cel mai mult timp
- Un calculator cu un set redus de registre necesită un timp ridicat pentru a gestiona apelurile de proceduri și revenirile
- Un principiu de proiectare RISC: asigurarea unui mijloc eficient de gestiune a procedurilor
  - Necesitatea existenței unui număr mare de registre

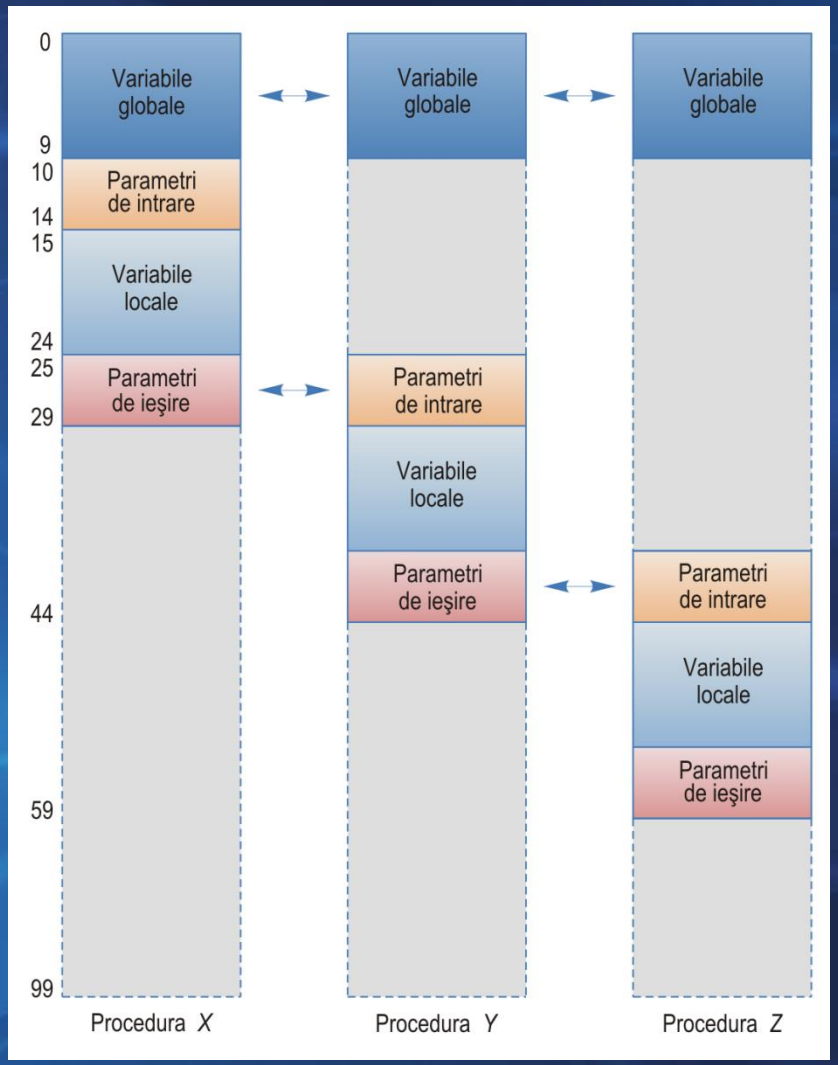
Ferestre suprapuse de registre (2)

- Conceptul *ferestrelor suprapuse de registre*
  - Setul de registre este împărțit în grupe de registre → ferestre
  - Registre globale, registre locale
  - Fiecărei proceduri i se asignează o fereastră separată în setul de registre
  - Baza ferestrei este adresată de pointerul ferestrei curente (*Current Window Pointer* – CWP)

Ferestre suprapuse de registre (3)

- Ferestrele de registre pot fi utile pentru transmiterea eficientă a parametrilor
  - Ferestrele de registre sunt suprapuse
  - Parametrii pot fi transmiși fără a modifica CWP $\rightarrow$ plasarea acestora în partea care se suprapune din cele două ferestre
  - Parametrii vor fi accesibili atât pentru procedura apelantă, cât și pentru procedura apelată

Ferestre suprapuse de registre (4)



Arhitecturi RISC

- Introducere
- Cauze ale complexității arhitecturale
- Avantaje ale arhitecturilor RISC
- Ferestre suprapuse de registre
- Caracteristici ale arhitecturilor RISC
- Comparație între arhitecturile RISC și CISC
- Arhitectura ARM

Caracteristici ale arhitecturilor RISC (1)

1. O arhitectură RISC este de tip *“load/store”*
2. Execuția majorității instrucțiunilor necesită un singur ciclu de ceas
3. Instrucțiunile au format fix
4. Unitatea de control este cablată
5. Număr redus de formate de instrucțiuni
6. UCP are un număr mare de registre
   - Alternativă: memorie *cache* de dimensiuni mari

Caracteristici ale arhitecturilor RISC (2)

7. Compilatorul are o complexitate ridicată
   - Rearanjează instrucțiunile
   - Tratează salturile întârziate
8. Există relativ puține instrucțiuni și foarte puține moduri de adresare
9. Facilitează operațiile limbajelor de nivel înalt
10. Utilizează sisteme *pipeline* de instrucțiuni și metode pentru rezolvarea problemei salturilor

Arhitecturi RISC

- Introducere
- Cauze ale complexității arhitecturale
- Avantaje ale arhitecturilor RISC
- Ferestre suprapuse de registre
- Caracteristici ale arhitecturilor RISC
- Comparație între arhitecturile RISC și CISC
- Arhitectura ARM

Comparație între arhitecturile RISC și CISC (1)

- Timpul de execuție al unui program poate fi influențat de:
  - Numărul de instrucțiuni din program
  - Numărul mediu de cicluri de ceas pe instrucțiune
  - Durata ciclului de ceas
- CISC: reduc numărul de instrucțiuni necesare într-un program $\rightarrow$ instrucțiuni speciale
- RISC: reduc numărul mediu al ciclurilor de ceas pe instrucțiune

Comparație între arhitecturile RISC și CISC (2)

- RISC și CISC: reduc durata ciclului de ceas
- RISC: sunt calculatoare de tip *"load/store"*
  - Pot obține un grad ridicat de concurență → separarea execuției instrucțiunilor de încărcare și memorare de alte instrucțiuni
- CISC: set de instrucțiuni de tip memorie-registru
  - Nu pot obține același grad de concurență ca și calculatoarele RISC

Comparație între arhitecturile RISC și CISC (3)

- Dezavantaje RISC
  - Instrucțiuni simple → performanțele depind de eficiența compilatorului
  - Alocarea registrelor este complexă → crește complexitatea compilatorului
  - Timpul de dezvoltare al programelor este mai lung
  - Programele RISC sunt mai lungi
- Proiectarea unui procesor RISC poate fi îmbunătățită prin utilizarea unor principii CISC

Arhitecturi RISC

- Introducere
- Cauze ale complexității arhitecturale
- Avantaje ale arhitecturilor RISC
- Ferestre suprapuse de registre
- Caracteristici ale arhitecturilor RISC
- Comparație între arhitecturile RISC și CISC
- Arhitectura ARM

Arhitectura ARM

- Arhitectura ARM
  - Prezentare generală ARM
  - Arhitectura ARMv7
  - Arhitectura ARMv8
    - Prezentare generală ARMv8
    - Starea de execuție AArch32
    - Starea de execuție AArch64

Prezentare generală ARM (1)

- ARM – *Advanced RISC Machine*
- Denumirea inițială: *Acorn RISC Machine*
- Proiectată la firma Acorn Computers (UK)
  - Primul procesor ARM comercial: ARM1 (1985), utilizat în calculatorul BBC Micro
  - Bazat pe procesorul Motorola 6502 (8 biți)
  - Modificări: arhitectură RISC de 32 de biți; lungime fixă a instrucțiunilor; arhitectură *load/store*

Prezentare generală ARM (2)

- Dezvoltarea procesoarelor ARM a continuat în firma Advanced RISC Machines Ltd. (1990)
  - Acorn Computers, VLSI Technology Inc., Apple Computer
- Primul procesor dezvoltat de ARM Ltd.: ARM6 (1992)
  - Au fost licențiate diferite versiuni ARM6: ARM610 (Apple), StrongARM (DEC), XScale (Intel)

Prezentare generală ARM (3)

- În 1998, ARM Ltd. a devenit ARM Holdings
  - A fost achiziționat de SoftBank Group (2016)
  - Urmează să fie achiziționat de NVIDIA (2020)
- Dezvoltarea procesoarelor ARM a continuat cu seriile ARM7 .. ARM11, iar apoi cu seriile ARM Cortex
- ARM Holdings proiectează arhitectura procesoarelor ARM și licențiază proiectele
- Procesoarele ARM se utilizează pe scară largă în: dispozitive mobile, sisteme înglobate, calculatoare portabile

Prezentare generală ARM (4)

- Nucleele ARM pot fi integrate în circuite SoC (*System-on-Chip*) ale diferiților producători
  - NVIDIA: Tegra (de ex., Tegra X2, Tegra Xavier)
  - Apple: A12 Bionic, A13 Bionic, A14 Bionic
  - Samsung: Exynos (de ex., Exynos 9820, Exynos 9825, Exynos 990)
  - Qualcomm: Snapdragon (de ex., Snapdragon 855, Snapdragon 865, Snapdragon 865+ 5G)

Prezentare generală ARM (5)

- Arhitectura ARM definește:
  - O mașină abstractă EP (element de procesare)
  - Reguli de utilizare ale EP
- Modelul de programare ARM
  - Registre de uz general
  - Moduri ale procesorului
  - Fiecare mod poate oferi diferite tipuri de accese la memorie și poate avea un subset propriu de registre

Arhitectura ARM

- Arhitectura ARM
  - Prezentare generală ARM
  - Arhitectura ARMv7
  - Arhitectura ARMv8
    - Prezentare generală ARMv8
    - Starea de execuție AArch32
    - Starea de execuție AArch64

Arhitectura ARMv7 (1)

- Sunt definite trei profiluri arhitecturale
  - Profilul A (*Application*): Cortex-A
    - Pentru sisteme care necesită performanțe ridicate: tablete, telefoane, aparate TV
  - Profilul R (*Real-time*): Cortex-R
    - Pentru sisteme înglobate cu performanțe ridicate: sisteme pentru automobile, medicale, industriale
  - Profilul M (*Microcontroller*): Cortex-M
    - Pentru sisteme înglobate cu consum și cost redus: electronica de consum, instrumente medicale

Arhitectura ARMv7 (2)

- Moduri ale EP
  - *User* (usr): mod neprivilegiat
  - *System* (sys): mod privilegiat
  - *Interrupt Request* (irq): întreruperi normale
  - *Fast Interrupt Request* (fiq): întreruperi rapide
  - *Supervisor* (svc): după resetare sau o instrucțiune de întrerupere software
  - *Abort* (abt): acces ilegal la memoria de date sau la cea de program (*Data/Prefetch Abort*)
  - *Undefined* (und): instrucțiune nedefinită

Arhitectura ARMv7 (3)

- Registre vizibile în modurile usr, sys: R0..R15
  - R13: *Stack Pointer* (SP)
  - R14: *Link Register* (LR)
  - R15: *Program Counter* (PC)
  - CPSR (*Current Program Status Register*)
- Pentru fiecare mod de excepție există un registru SPSR (*Saved Program Status Register*) și copii ale registrelor SP, LR
- În modul fiq: copii ale registrelor R8..R12

Arhitectura ARMv7 (4)

- Extensia Thumb
  - Subset al instrucțiunilor ARM
  - Instrucțiunile sunt comprimate pe 16 biți și sunt decomprimate pe 32 de biți înainte de execuție
  - Comutarea între starea ARM și starea Thumb: cu instrucțiunea BX (*Branch and eXchange*)
  - Există limitări ale setului de instrucțiuni Thumb

Arhitectura ARMv7 (5)

- Extensia Thumb-2
  - Noi instrucțiuni de 32 de biți
  - Set de instrucțiuni extins: operații pe câmpuri de biți, execuție condițională
  - Se pot genera instrucțiuni Thumb sau ARM din același cod sursă → limbaj de asamblare unificat (UAL – *Unified Assembly Language*)
  - Performanțe superioare față de setul Thumb: dimensiunea memoriei redusă cu 30%; performanța îmbunătățită cu până la 38%

Arhitectura ARMv7 (6)

- Extensia DSP (*Digital Signal Processor*)
  - Adunarea și scăderea utilizează aritmetica saturată cu semn
  - Noi instrucțiuni de înmulțire și înmulțire cu acumulare (MAC – *Multiply and Accumulate*), cu semn
- Extensia VFP (*Vector Floating Point*)
  - Definită pentru profilurile A și R
  - Instrucțiuni în virgulă mobilă cu precizie simplă și precizie dublă
  - Versiunile 3 (VFPv3) și 4 (VFPv4)

Arhitectura ARMv7 (7)

- Versiunea 4: date de 16 biți; instrucțiuni MAC fuzionate
- 32 registre de 64 de biți (VFPvx-D32), sau
- 16 registre de 64 de biți (VFPvx-D16)

- Extensia ARM NEON

  - Set de instrucțiuni SIMD (*Single Instruction, Multiple Data*) avansate
  - Accelerare pentru aplicații multimedia și DSP
    - Codificare/decodificare video și audio, grafică 2D/3D, sinteza vocii, prelucrarea imaginilor

Arhitectura ARMv7 (8)

- 32 de registre x 64 de biți
  - Pot fi utilizate ca 16 registre x 128 de biți
  - Datele din registre sunt considerate ca vectori cu elemente de același tip
- Tipuri de date: întregi (8, 16, 32 sau 64 biți), în VM cu precizie simplă
- Avantaje:
  - Suport pentru diferite standarde: MPEG-4, H.264, Real, AVS
  - Se poate utiliza pentru prelucrare vectorială
  - Performanță crescută cu 60..150% pentru codec-uri video și de 4..8 ori pentru algoritmi DSP simpli

Arhitectura ARMv7 (9)

- Extensia de securitate TrustZone

  - Arhitectură hardware care permite rularea unui *kernel* considerat sigur
    - TEE (*Trusted Execution Environment*)
  - Resursele hardware și software se pot afla în una din două stări (domenii):
    - Starea sigură: subsistemul de securitate
    - Starea normală: restul sistemului
  - Procesoarele TrustZone conțin două nuclee virtuale pentru cele două stări
  - Comutarea contextului între nuclee: printr-un nou mod al EP, *Monitor* (mon)

Arhitectura ARMv7 (10)

- Extensia de virtualizare ARM
  - Suport hardware pentru rularea unor SO pe mașini virtuale (MV)
  - Permite rularea codului binar nativ al unui SO într-o MV cu degradarea minimă a performanțelor
  - Monitorul MV: hipervizor
  - Se introduce un nou mod al EP, *Hypervizor* (hyp), cu un nivel de privilegiu mai ridicat decât modurile *kernel* existente

Arhitectura ARM

- Arhitectura ARM
  - Prezentare generală ARM
  - Arhitectura ARMv7
  - Arhitectura ARMv8
    - Prezentare generală ARMv8
    - Starea de execuție AArch32
    - Starea de execuție AArch64

Prezentare generală ARMv8 (1)

- Se păstrează profilurile arhitecturale introduse în versiunea ARMv7: A, R, M
- Se va descrie doar profilul ARMv8-A
- S-au păstrat majoritatea extensiilor din versiunile precedente
  - Extensiile NEON și VFP fac parte din arhitectura ARMv8-A de bază
  - Extensie criptografică opțională pentru algoritmul AES (*Advanced Encryption Standard*) și funcțiile SHA (*Secure Hash Algorithm*) SHA-1, SHA-256

Prezentare generală ARMv8 (2)

- Stare de execuție: definește mediul de execuție pentru un EP
- Două stări de execuție, AArch32 și AArch64
  - AArch32: stare de execuție de 32 de biți
    - Compatibilă cu versiunile precedente
    - Setul de instrucțiuni A32 (ARM)
    - Setul de instrucțiuni T32 (Thumb-2)
  - AArch64: stare de execuție de 64 de biți
    - Adrese de 64 biți, operanzi de 32 sau 64 biți
    - Nou set de instrucțiuni, A64

Prezentare generală ARMv8 (3)

- Modelul excepțiilor
  - AArch32
    - Același model ca și în arhitectura ARMv7
    - Bazat pe modurile EP
  - AArch64
    - Bazat pe patru nivele de excepție, EL0 .. EL3
    - Nivelul EL0: neprivilegiat
    - Nivelele EL1, EL2, EL3: privilegiate
- Implementări: Cortex-A5x, Cortex-A7x, Apple Ax, Qualcomm Snapdragon 84x/85x/86x

Arhitectura ARM

- Arhitectura ARM
  - Prezentare generală ARM
  - Arhitectura ARMv7
  - Arhitectura ARMv8
    - Prezentare generală ARMv8
    - Starea de execuție AArch32
    - Starea de execuție AArch64

Starea de execuție AArch32 (1)

- Moduri ale EP
  - Fiecare mod este implementat la un anumit nivel de excepție
  - User (*usr*): execuție neprivilegiată (EL0)
    - Programele executate în acest mod nu pot accesa resurse sau zone de memorie protejate
  - FIQ (*fiq*): tratarea unei cereri de întrerupere rapidă (EL1 sau EL3)
  - IRQ (*irq*): tratarea unei cereri de întrerupere normală (EL1 sau EL3)
  - Supervisor (*svc*): tratarea excepției *Supervisor Call* $\rightarrow$ instrucțiunea SVC (EL1 sau EL3)

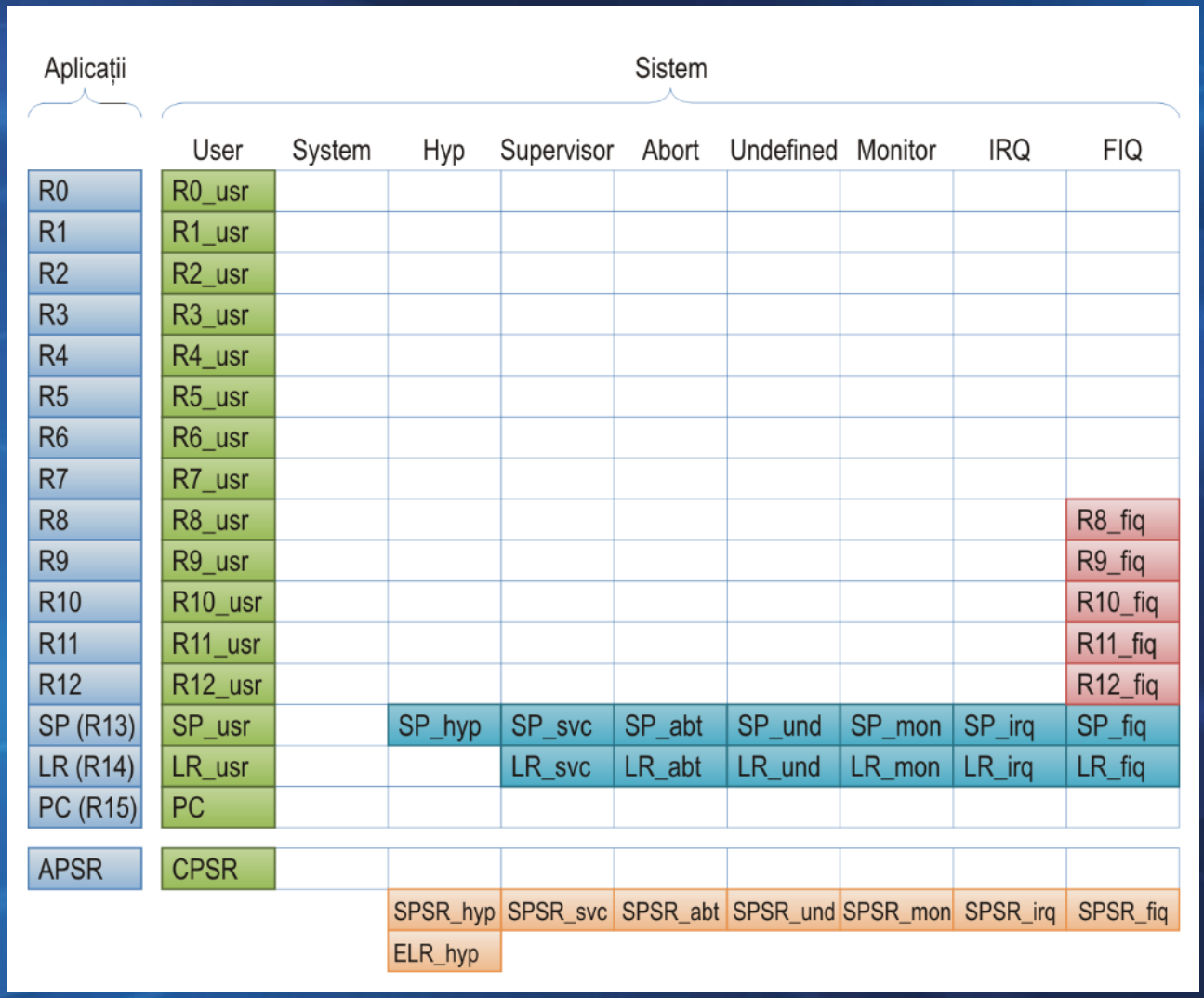
Starea de execuție AArch32 (2)

- Monitor (*mon*): tratarea excepției *Secure Monitor Call* → instrucțiunea SMC (EL3)
- Abort (*abt*): tratarea excepției *Data Abort* sau *Prefetch Abort* (EL1 sau EL3)
- Hyp (*hyp*): tratarea excepției *Hypervisor Call* → instrucțiunea HVC (EL2)
- Undefined (*und*): tratarea excepției de instrucțiune nedefinită (EL1 sau EL3)
- System (*sys*): nu apare în urma unei excepții → scrierea biților de mod din registrul CPSR

Starea de execuție AArch32 (3)

- Registre disponibile
  - Pentru aplicații
    - Registrele generale R0 .. R14
      - R13: SP (*Stack Pointer*)
      - R14: LR (*Link Register*)
    - Registre speciale: R15 (PC – *Program Counter*); APSR (*Application Program Status Register*)
  - Pentru sistem
    - Copii ale unor registre: SP (de ex., SP_svc), LR (de ex., LR_fiq), CPSR (de ex., SPSR_irq)

Starea de execuție AArch32 (4)



Aplicații	Sistem								
	User	System	Hyp	Supervisor	Abort	Undefined	Monitor	IRQ	FIQ
R0	R0_usr								
R1	R1_usr								
R2	R2_usr								
R3	R3_usr								
R4	R4_usr								
R5	R5_usr								
R6	R6_usr								
R7	R7_usr								
R8	R8_usr								R8_fiq
R9	R9_usr								R9_fiq
R10	R10_usr								R10_fiq
R11	R11_usr								R11_fiq
R12	R12_usr								R12_fiq
SP (R13)	SP_usr		SP_hyp	SP_svc	SP_abt	SP_und	SP_mon	SP_irq	SP_fiq
LR (R14)	LR_usr			LR_svc	LR_abt	LR_und	LR_mon	LR_irq	LR_fiq
PC (R15)	PC								
APSR	CPSR								
			SPSR_hyp	SPSR_svc	SPSR_abt	SPSR_und	SPSR_mon	SPSR_irq	SPSR_fiq
			ELR_hyp						

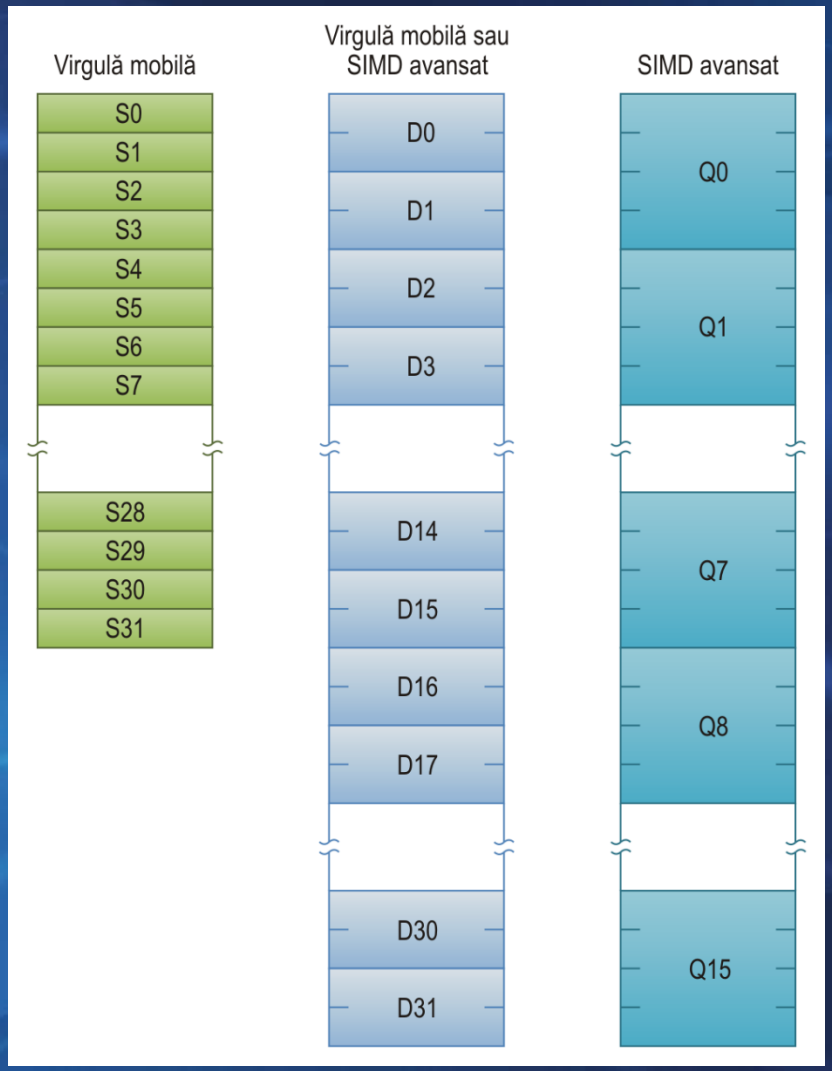
Starea de execuție AArch32 (5)

- Registre pentru instrucțiuni SIMD și în VM
  - Set de registre separat de registrele generale
  - Instrucțiuni SIMD avansate: operații asupra elementelor vectorilor
    - Un registru poate conține elemente de același tip (întregi sau în VM cu precizie simplă)
    - Numărul de elemente depinde de tipul lor și dimensiunea registrului
  - Instrucțiuni în VM: operații scalare în VM cu precizie simplă sau dublă

Starea de execuție AArch32 (6)

- Set comun de registre (SIMD și VM): 32 registre de 64 biți
- Vederi diferite ale registrelor pentru instrucțiunile SIMD avansate și cele în VM
- Instrucțiuni SIMD avansate:
  - 32 registre de 64 biți, D0 .. D31, sau
  - 16 registre de 128 biți, Q0 .. Q15
- Instrucțiuni în VM:
  - 32 registre de 32 biți, S0 .. S31, sau
  - 32 registre de 64 biți, D0 .. D31

Starea de execuție AArch32 (7)



Arhitectura ARM

- Arhitectura ARM
  - Prezentare generală ARM
  - Arhitectura ARMv7
  - Arhitectura ARMv8
    - Prezentare generală ARMv8
    - Starea de execuție AArch32
    - Starea de execuție AArch64

Starea de execuție AArch64 (1)

- 31 registre generale: R0 .. R30
  - Pot fi accesate ca registre de 64 biți (X0 .. X30) sau ca registre de 32 de biți (W0 .. W30)
  - X30: registru de legătură
  - ZR: registru zero → nu este un registru fizic
- SP: indicator de stivă dedicat (64 biți)
  - Cuvântul inferior poate fi accesat prin numele WSP
- PC: contor de program (64 biți)
  - Nu poate fi înscris prin program

Starea de execuție AArch64 (2)

- Set comun de registre pentru instrucțiuni SIMD avansate și în VM
  - 32 registre de 128 biți, V0 .. V31
  - Registrele pot fi accesate ca și:
    - Cuvinte cvadruple (128 biți), Q0 .. Q31
    - Cuvinte duble (64 biți), D0 .. D31
    - Cuvinte (32 biți), S0 .. S31
    - Semi-cuvinte (16 biți), H0 .. H31
    - Octeți, B0 .. B31

Rezumat (1)

- Principiul arhitecturilor RISC: adăugarea la setul de instrucțiuni doar a acelor instrucțiuni care determină un câștig de performanță
- Arhitecturile RISC au mai multe avantaje față de arhitecturile CISC:
  - Viteza mai ridicată: utilizarea mai eficientă a sistemului *pipeline* de instrucțiuni; unitate de control cablată; număr mare de registre
  - Timpul de proiectare mai redus
  - Facilitarea utilizării limbajelor de nivel înalt

Rezumat (2)

- Conceptul ferestrelor suprapuse de registre permite transmiterea eficientă a parametrilor între procedura apelantă și cea apelată
- Se poate observa influența reciprocă între arhitecturile RISC și CISC
- Numeroase procesoare sunt bazate pe arhitectura ARM
  - Utilizate pe scară largă în dispozitive mobile și sisteme înglobate
  - Nucleele ARM sunt integrate în circuite SoC

Rezumat (3)

- Arhitectura ARM definește o mașină abstractă (EP) și reguli de utilizare ale acesteia
  - Modelul de programare ARM definește diferite moduri ale EP și registrele disponibile
- Arhitectura ARMv7 introduce profilurile A, R, M
  - Extensii arhitecturale: Thumb, Thumb-2, DSP, VFP, ARM NEON, ARM TrustZone, virtualizare
- Arhitectura ARMv8 definește două stări de execuție, AArch32 (32 biți) și AArch64 (64 biți)
  - AArch64 introduce un nou model al excepțiilor bazat pe patru nivele de excepție

Noțiuni, cunoștințe (1)

- Principiul calculatoarelor cu set redus de instrucțiuni (RISC)
- Avantaje ale arhitecturilor RISC
- Principiul ferestrelor suprapuse de registre
- Caracteristici ale arhitecturilor RISC
- Comparație între arhitecturile RISC și cele CISC
- Profiluri definite de arhitectura ARMv7
- Extensiile ARMv7 Thumb și Thumb-2
- Extensia ARM NEON

Noțiuni, cunoștințe (2)

- Extensia de securitate ARM TrustZone
- Extensia de virtualizare ARM
- Stările de execuție AArch32 și AArch64 definite de arhitectura ARMv8
- Modelul excepțiilor în stările de execuție AArch32 și AArch64
- Moduri ale EP în starea de execuție AArch32
- Registre disponibile în starea de execuție AArch64