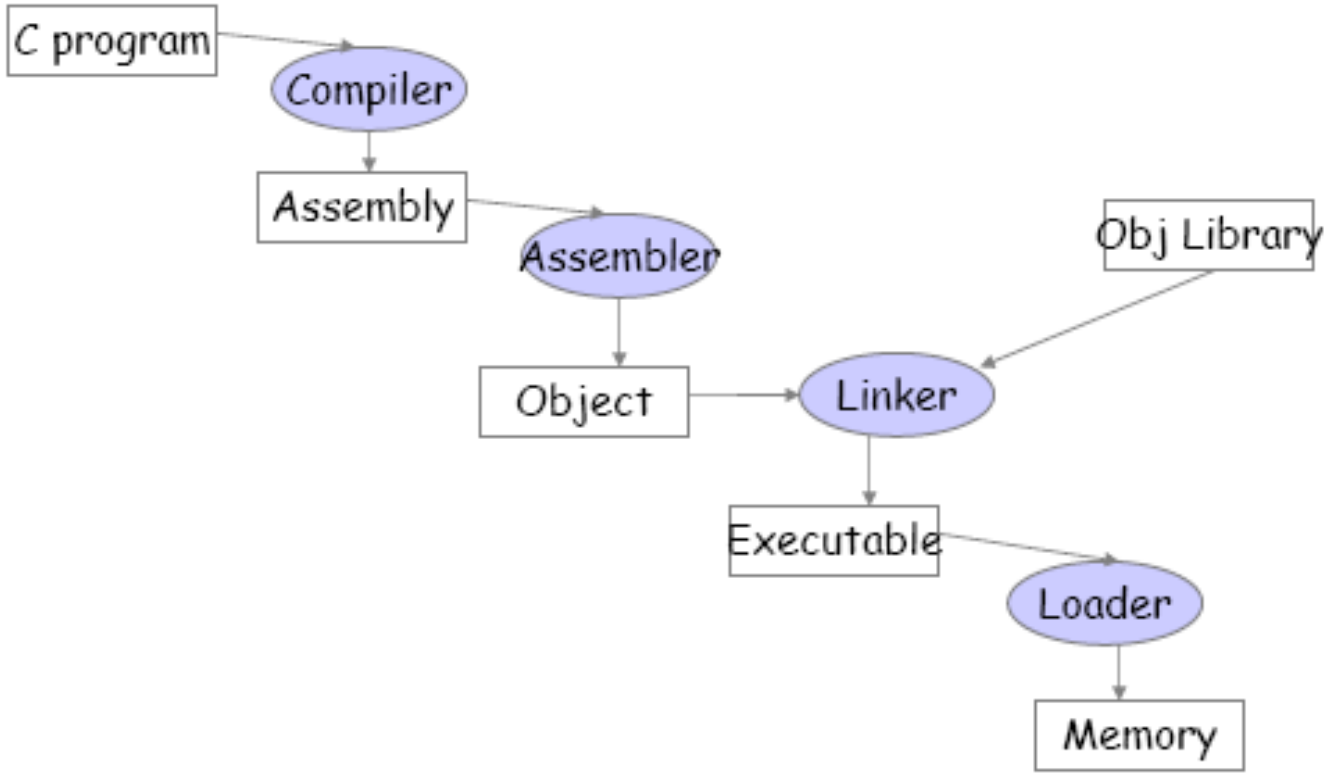
Αρχιτεκτονική του ARM

K.Harteros – G.Kornaros

Περίγραμμα

- Περιγραφή της αρχιτεκτονικής του ARM από την οπτική γωνία του προγραμματιστή
  - ✓ Processor modes
  - ✓ General purpose registers
  - ✓ Special purpose registers
  - ✓ Exception handling
- Σετ εντολών του ARM
  - ✓ Data processing instructions
  - ✓ Conditional execution of instructions
  - ✓ Barrel shifter
- Ανακοινώσεις

Μετάφραση προγράμματος



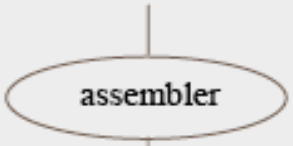
Κώδικας Μηχανής

- Κώδικας σε υψηλού επιπέδου γλώσσα (C)

```
void swap(int a,int b)
{
   tmp = a;
   a = b;
   b = tmp;
} /* end swap() */
```

- Πρόγραμμα σε assembly γλώσσα

```
MOV     r2,r0
MOV     r0,r1
MOV     r1,r2
MOV     pc,r14
```



- Κώδικας σε γλώσσα μηχανής

```
0xe1a02000
0xe1a00001
0xe1a01002

0xe1a0f00e
```

Μετάφραση σε δυαδικό

- Assembler
  - ✓ Μετατρέπει την assembly σε object code για κάθε module (source file)
  - ✓ Symbol table for undefined/extern references
- Linker
  - ✓ Ψάχνει τις βιβλιοθήκες του προγράμματος για να βρει ρουτίνες βιβλιοθηκης που χρησιμοποιούνται από το πρόγραμμα
  - ✓ Επιλύει εξωτερικές αναφορές
- Loader
  - ✓ Φορτώνει το εκτελέσιμο στην μνήμη
  - ✓ Copies instructions and data from executable to memory
  - ✓ Copy arguments to the program into stack
  - ✓ Jumps to a start-up routine which copies program arguments from stack to registers and calls programs main routine

Περιοχές ενός προγράμματος

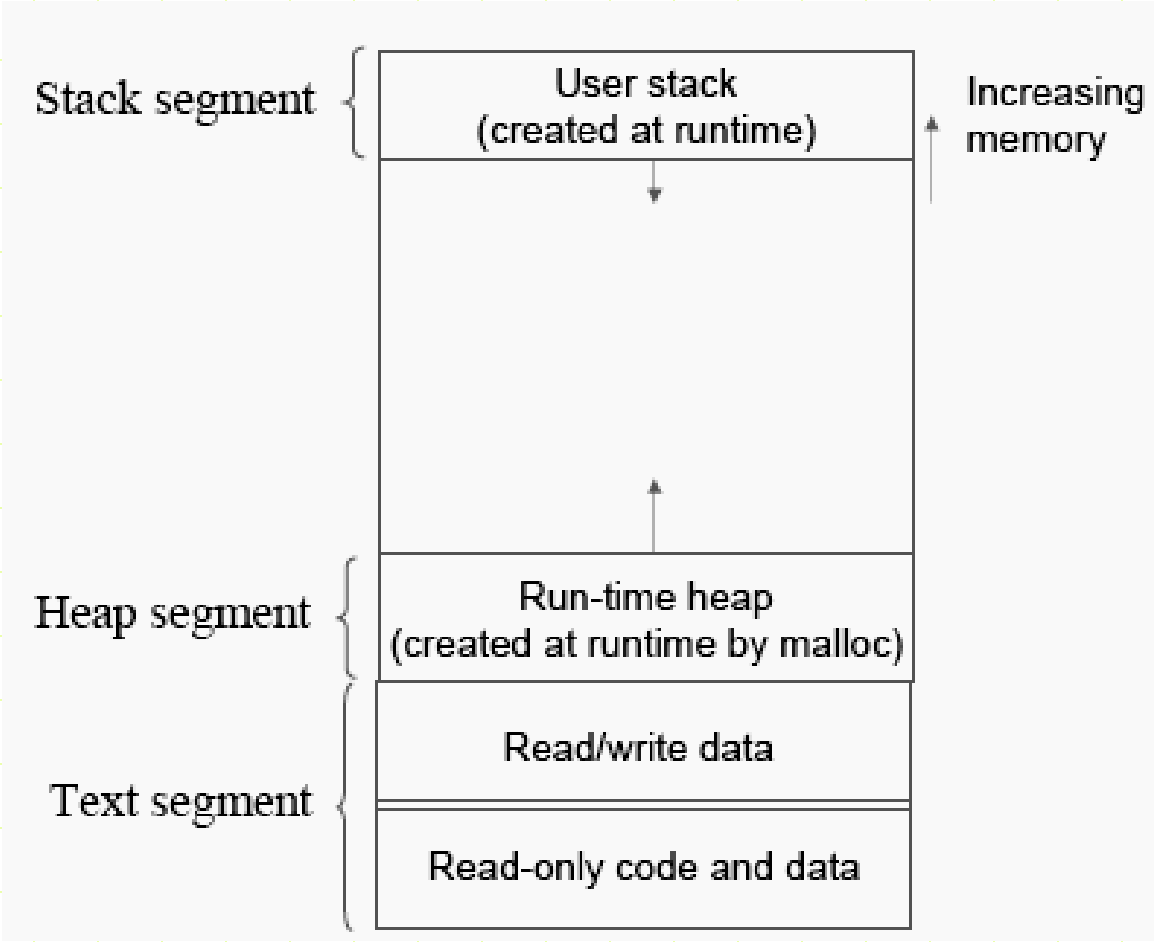
- Όταν ένα πρόγραμμα φορτώνεται στην μνήμη οργανώνεται μέσα σε 3 περιοχές της μνήμης που ονομάζονται :
- *Segments*
  - ✓ Text segment, stack segment, and heap segment

- Text segment (sometimes also called the code segment) is where the compiled code of the program itself resides

- Stack segment
  - ✓ Memory is allocated for automatic variables within functions
  - ✓ Particularly useful when subroutines are invoked
  - ✓ The variables and data associated with nested routines are stored on the stack
  - ✓ Used and re-used during a program's execution

- Heap segment
  - ✓ Provides more stable storage of data for a program
  - ✓ Heap-allocated memory remains in existence for the duration of a program
  - ✓ Typically holds global and static variables

Περιοχές μνήμης ενός προγράμματος



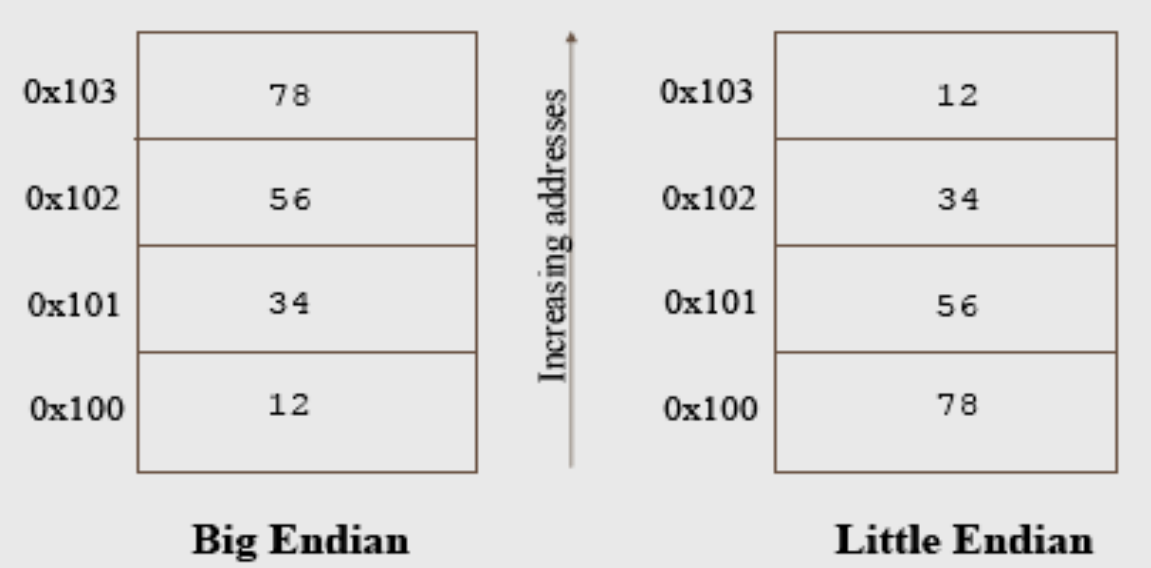
Διάδοση του ARM

Μέγεθος Δεδομένων και Εντολών του ARM

- Ο ARM είναι 32-bit RISC architecture
- Όταν μιλάμε με βάση τον ARM: -- ΟΡΙΣΜΟΙ --
  - ✓ **Byte** means 8 bits
  - ✓ **Halfword** means 16 bits (two bytes)
  - ✓ **Word** means 32 bits (four bytes)
- Πολλοί επεξεργαστές ARM υλοποιούν δύο σετ εντολών
  - ✓ 32-bit ARM Instruction Set
  - ✓ 16-bit Thumb Instruction Set
- Μπορεί να γίνει configured να βλέπει τις λέξεις που είναι αποθηκευμένες στην μνήμη είτε ως Big-endian είτε ως Little-endian format
- Quiz
  - ✓ Που/Γιατί θα μπορούσαν να χρησιμοποιηθούν οι εντολές 16-bit Thumb εντολές?

Big <> Little Endian

- Πως αποθηκεύεται η λέξη 0x12345678 στην μνήμη ?



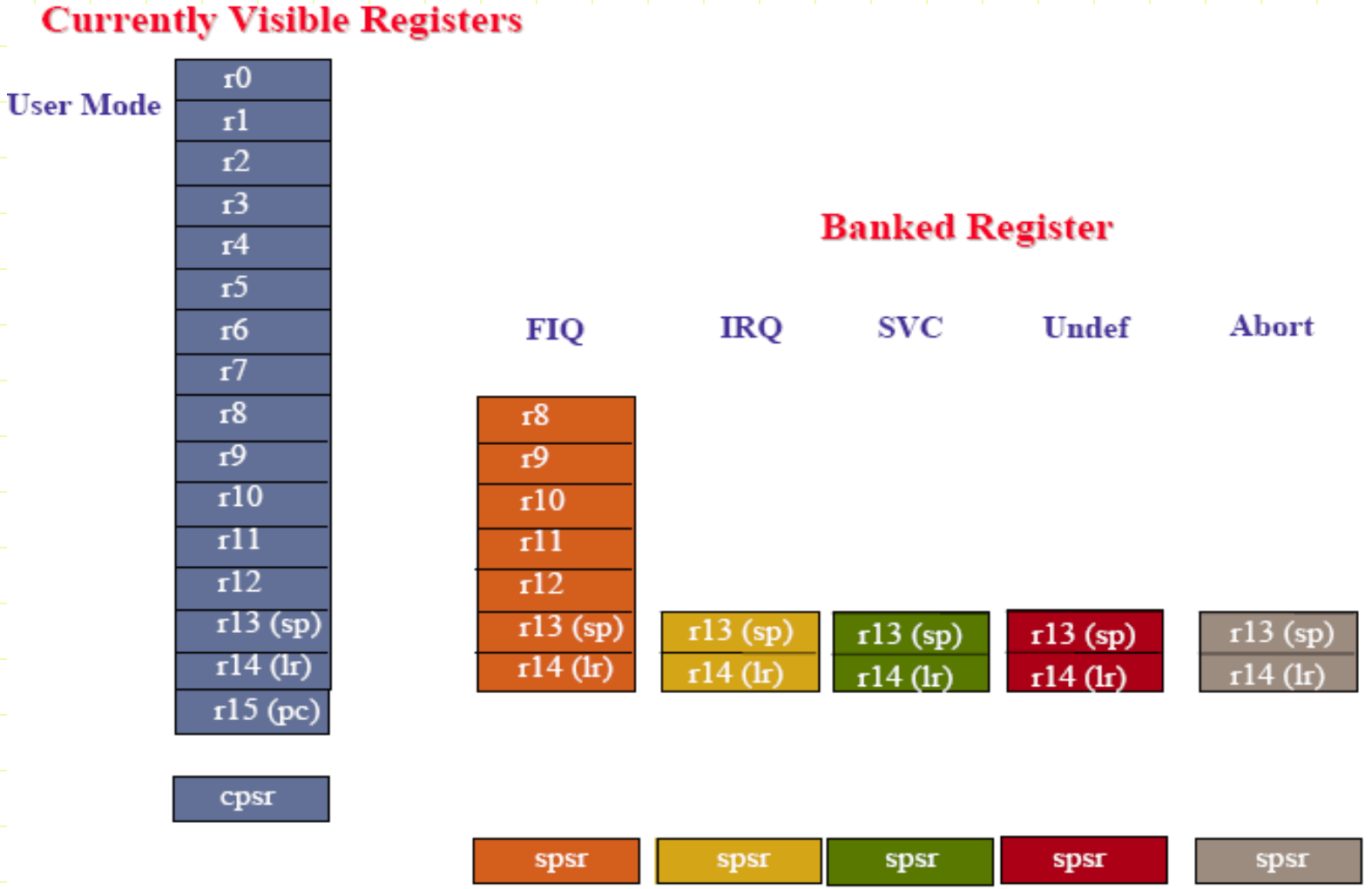
ARM Programmer's Model

- Τι είναι το μοντέλο του προγραμματιστή ?
- Ο ARM έχει επτά βασικά modes λειτουργίας :
  - ✓ **User** : normal program execution mode (unprivileged mode)
  - ✓ **FIQ** : used for handling a fast interrupt (privileged mode)
  - ✓ **IRQ** : used for handling a low priority (normal) interrupt (privileged mode)
  - ✓ **Supervisor** : entered on reset and when a Software Interrupt instruction is executed (privileged mode)
  - ✓ **Abort** : used for handling memory access violations (privileged mode)
  - ✓ **Undefined** : used for handling undefined instructions (privileged mode)
  - ✓ **System** : a privileged mode that uses the same registers as the user mode (privileged mode)

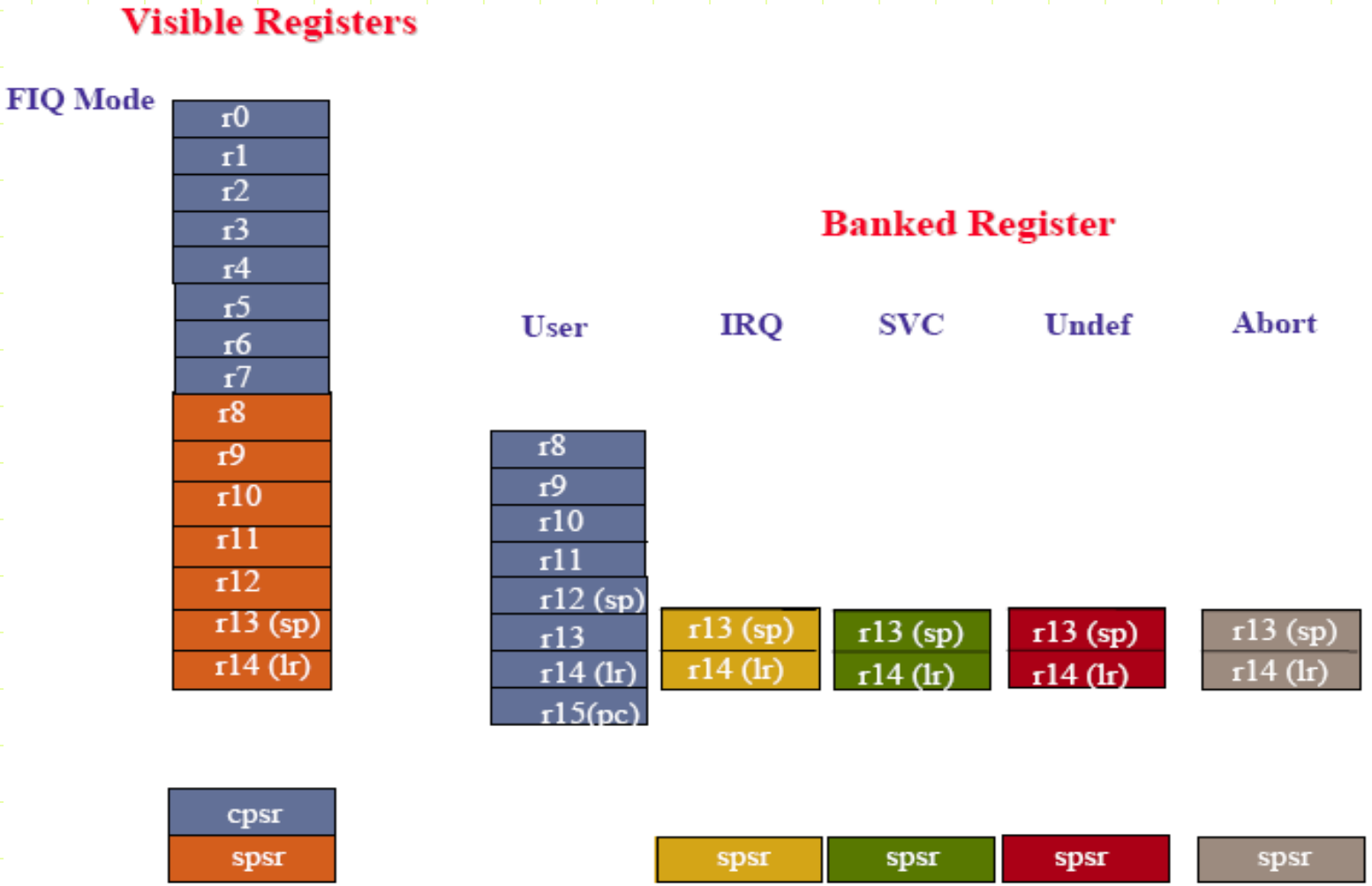
Οι καταχωρητές του ARM

- Ο ARM έχει συνολικά 37 registers και όλοι είναι 32-bits
  - ✓ 30 general purpose registers
  - ✓ 1 dedicated program counter (pc)
  - ✓ 1 dedicated current program status register (cpsr)
  - ✓ 5 dedicated saved program status registers (spsr)
- Σε οποιοδήποτε mode, μόνο ένα υποσύνολο των 37 registers είναι ορατοί
  - ✓ The hidden registers are called **banked registers**
  - ✓ Το τρέχον mode του επεξεργαστή ορίζει ποιοι καταχωρητές είναι προσπελάσιμοι
- Κάθε mode μπορεί να προσπελάσει:
  - ✓ A particular set of r0-r12 registers
  - ✓ A particular r13 (the stack pointer, sp) and r14 (the link register, lr)
  - ✓ The program counter, r15 (pc)
  - ✓ The current program status register, cpsr
- Τα Προνομιούχα modes (εκτός System) μπορούν να έχουν πρόσβαση σε
  - ✓ A particular saved program status register (spsr)

Οι καταχωρητές του ARM (user mode)



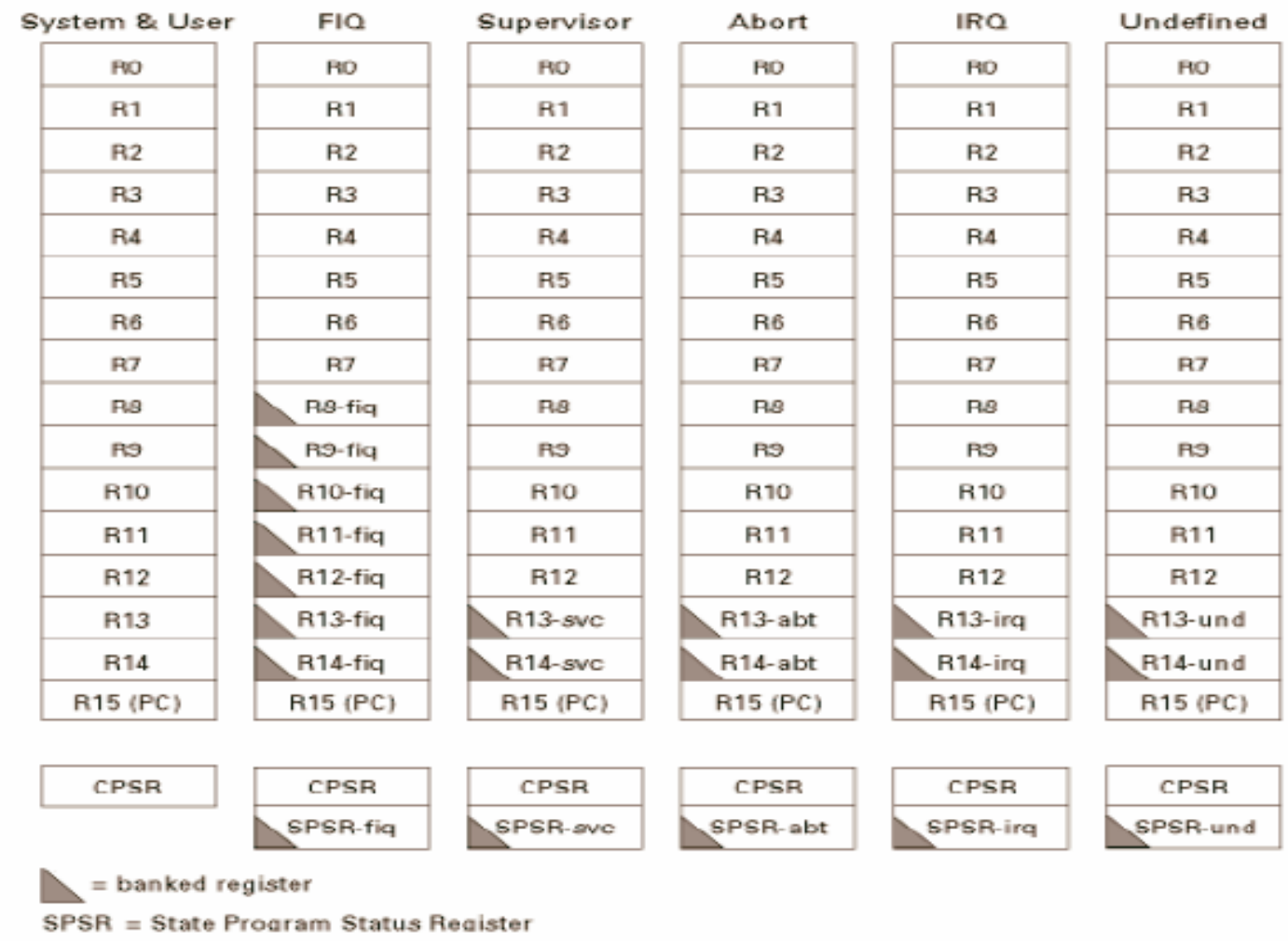
Οι καταχωρητές του ARM (FIQ mode)



Banked καταχωρητές

- Banking of registers σημαίνει
  - ✓ Ένας συγκεκριμένος καταχωρητής εξαρτάται όχι μόνο από τον αριθμό του (r0, r1, r2 … r15), αλλά επίσης και από το mode του επεξεργαστή
- Οι τιμές που είναι αποθηκευμένες στους banked καταχωρητές διατηρούνται όταν αλλάζουμε mode.
- Example – assume that the processor is executing in user mode
  - ✓ In user mode, assume that the processor writes **0** in r0 and **8** in r8
  - ✓ Processor changes to FIQ mode
    - In FIQ mode, the value of r0 is
    - If processor overwrites both r0 and r8 with **1** in FIQ mode and changes back to user mode
  - ✓ The new value stored in r0 (in user mode) is
  - ✓ The new value stored in r8 (in user mode) is

Οργάνωση των καταχωρητών του ARM



Εξαιρέσεις

- Οι εξαιρέσεις στον ARM αναφέρονται σε οποιαδήποτε συνθήκη που μπορεί να σταματήσει την κανονική ακολουθιακή εκτέλεση των εντολών ενός προγράμματος

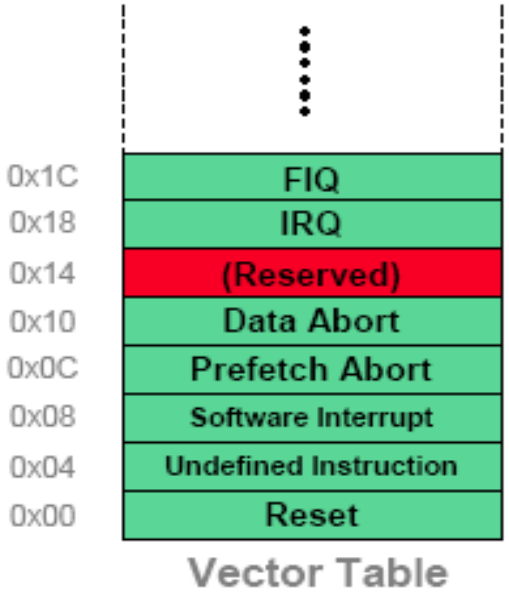
Exception	*Mode*	*Description*
Reset	Supervisor	Occurs when the processor's reset button is asserted. This exception is only expected to occur for signaling power up or for resetting the processor. A soft reset can be achieved by branching to reset vector 0x00000000
Undefined Instruction	Undef	Occurs if neither the processor, nor any of the coprocessors, recognize the currently executing instruction
Software Interrupt	Supervisor	This is a user-defined synchronous interrupt. It allows a program running in the User mode to request privileged operations (for example an RTOS function) that run in Supervisor mode
Prefetch Abort	Abort	Occurs when a processor attempts to execute an instruction that was not fetched, because the address was illegal
Data Abort	Abort	Occurs when a data transfer instruction attempts to load or store data at an illegal address
IRQ	IRQ	Occurs when the processor's external interrupt request pin is asserted and the I bit in the `cpsr` is clear
FIQ	FIQ	Occurs when the processor's external fast interrupt request pin is asserted and the I bit in the `cpsr` is clear

Χειρισμός Εξαιρέσεων

- Χειριστής Εξαιρέσεων (Exception Handler)
  - ✓ Most exceptions have an associated software exception handler that executes when that particular exception occurs
- Where is this exception handler located?
- Πίνακας Διανυσμάτων (Vector table)
  - ✓ Reserved area of 32 bytes at the end of the memory map (starting at address 0x0)
  - ✓ One word of space for each exception type
  - ✓ Contains a Branch or Load PC instruction for the exception handler
- Exception modes and registers
  - ✓ Handling exceptions changes program from user to non-user mode
  - ✓ Each exception handler has access to its own set of registers
    - o Its own r13 (stack pointer)
    - o Its own r14 (link register)
    - o Its own spsr (Saved Program Status Register)
  - ✓ Exception handlers must save (restore) other register on entry (exit)

Χειρισμός Εξαιρέσεων

- Όταν συμβεί μια εξαίρεση, τότε ο ARM:
  - ✓ Copies cpsr into spsr_<mode>
  - ✓ Sets appropriate cpsr bits
- Αλλαγή σε ARM state
- Αλλαγή σε exception mode
- Απενεργοποίηση των interrupts
  - ✓ Stores the return address in lr_<mode>
  - ✓ Sets pc to vector address



- Για να επιστρέψει , ο exception handler πρέπει να:
  - ✓ Restore cpsr from spsr_<mode>
  - ✓ Restore pc from lr_<mode>

Τι συμβαίνει όταν οι Exceptions συμβούν ταυτόχρονα ?

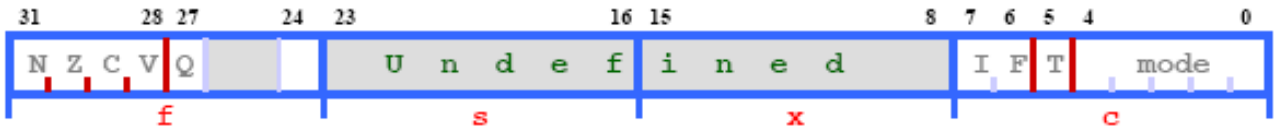
Vector address	Exception type	Exception mode	Priority (1=high, 6=low)
0x0	Reset	Supervisor (SVC)	1
0x4	Undefined Instruction	Undef	6
0x8	Software Interrupt (SWI)	Supervisor (SVC)	6
0xC	Prefetch Abort	Abort	5
0x10	Data Abort	Abort	2
0x14	*Reserved*	*Not applicable*	*Not applicable*
0x18	Interrupt (IRQ)	Interrupt (IRQ)	4
0x1C	Fast Interrupt (FIQ)	Fast Interrupt (FIQ)	3

Μετρητής προγράμματος

- Όταν ο επεξεργαστής εκτελεί κώδικα σε ARM state:
  - ✓ Όλες οι εντολές έχουν πλάτος 32 bits
  - ✓ Όλες οι εντολές πρέπει να είναι word aligned
  - ✓ Γι'αυτό το λόγο η τιμή του pc αποθηκεύεται στα bits [31:2] with bits [1:0] undefined (as instruction cannot be half-word or byte aligned)
- Όταν ο επεξεργαστής εκτελεί κώδικα σε Thumb state:
  - ✓ Όλες οι εντολές έχουν πλάτος 16 bits
  - ✓ Όλες οι εντολές πρέπει να είναι halfword aligned
  - ✓ Γι'αυτό το λόγο η τιμή του pc αποθηκεύεται στα bits [31:1] with bit [0] undefined (as instruction cannot be byte aligned)

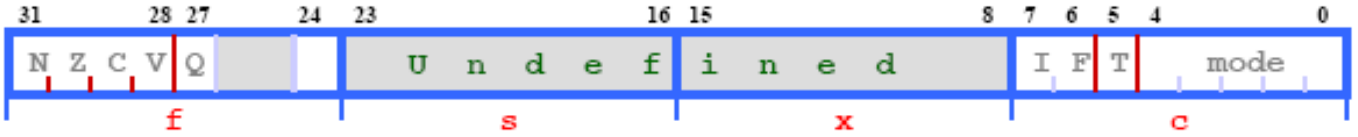
Current Program Status Register

- Ο Current Program Status Register (cpsr) είναι ένας εξειδικευμένος καταχωρητής ο οποίος:
  - ✓ Holds information about the most recently performed ALU operation
  - ✓ Controls the enabling and disabling of interrupts (both IRQ and FIQ)
  - ✓ Sets the processor operating mode
  - ✓ Sets the processor state



- cpsr can be divided into four fields, with each field being 8-bit wide
  - ✓ Flags field: contains the condition flags
  - ✓ Status field: reserved for future use
  - ✓ Extension field: reserved for future use
  - ✓ Control field: contains the processor mode, state and interrupt mask bits
- All fields of the cpsr can be read/written in privileged modes
- Only the flag field of cpsr can be written in User mode, all fields can be read in User mode
- Notation: we will use cpsr_f to refer to the flag fields of cpsr, cpsr_c to refer to control fields of cpsr, cpsr_cf to refer to both the flag and control fields of the cpsr …

Current Program Status Register (συν...)



- **Condition code flags**
  - ✓ **N = Negative result from ALU**
  - ✓ **Z = Zero result from ALU**
  - ✓ **C = ALU operation Carried out**
  - ✓ **V = ALU operation oVerflowed**
- **Sticky Overflow flag - Q flag**
  - ✓ **Architecture 5TE/J only**
  - ✓ **Indicates if saturation has occurred**
- **Interrupt Disable bits.**
  - ✓ **I = 1: Disables the IRQ.**
  - ✓ **F = 1: Disables the FIQ.**
- **T Bit**
  - ✓ **Architecture xT only**
  - ✓ **T = 0: Processor in ARM state**
  - ✓ **T = 1: Processor in Thumb state**
- **Mode bits**
  - ✓ **Specify the processor mode**

Υπενθύμιση : όταν συμβεί μία εξαίρεση τότε ο cpsr αντιγράφεται στο αντίστοιχο spsr_<mode>

Εισαγωγή στην Assembly γλώσσα του ARM

- Βασικά σημεία του ARM Instruction Set
- – αρχιτεκτονική “Load/Store”
- – 32-bit instructions (except when the processor is in Thumb state)
- – 32-bit and 8-bit data types
- – 32-bit addresses
- – Load and store multiple instructions
- – No instruction to move a 32-bit constant to a register (why?)
- – Conditional execution of all instructions
- – Co-processor instructions (we will *not* use these)

Κατηγορίες των εντολών του ARM

- ARM instructions can be broadly classified as
  - ✓ Data Processing Instructions: επεξεργασία δεδομένων μέσα στους registers
  - ✓ Branch Instructions: αλλαγή της ροής των εντολών ή κλήση υπορουτίνας.
  - ✓ Load-Store Instructions: μεταφορά δεδομένων μεταξύ registers και μνήμης
  - ✓ Software Interrupt Instruction: προκαλεί μία διακοπή interrupt
  - ✓ Program Status Instructions: διάβασμα/γράψιμό των καταχωρητών κατάστασης του επεξεργαστή

Ανατομία μίας εντολής ARM Data Processing

- **Operation** – Specifies the instruction to be performed
  Almost all ARM instructions can be conditionally executed
- **Cond** – specify the optional conditional flags which have to be set under which to execute the instruction
- **S bit** – Signifies that the instruction updates the conditional flags
- **Rd** – Specifies the destination register
- **Rn** – Specifies the first source operand register
- **ShifterOperand2** – Specifies the second source operand
  - ✓ Could be a register, immediate value, or a shifted register/immediate value
- Some data processing instructions may not specify the destination register or the source register

Εντολές επεξεργασίας Δεδομένων

- Αποτελούνται από :
  - Arithmetic: ADD ADC SUB SBC RSB RSC
  - Logical: AND ORR EOR BIC
  - Comparisons: CMP CMN TST TEQ
  - Data movement: MOV MVN
- παραδείγματα :

```
- ADD r0, r1, r2      %r0 ⇦ r1+r2
- ADS r0, r1, r2      %r0 ⇦ r1+r2 and update cpsr's nzcv bits
- ADC r0, r1, r2      %r0 ⇦ r1+r2+C C is the carry bit of cpsr
- SUB r2, r3, #1      %r2 ⇦ r3-1
- SBC r2, r3, r0      %r2 ⇦ r3-r0+C-1
- RSB r1, r2, r3      %r1 ⇦ r2-r3
- MOV r1, #0x8f       %r1 ⇦ 0x8f
- MVN r1, #0x8f       %r1 ⇦ 0xffffff70 (r1 := not 0x8f)
- CMP r0, r1          %compute r0-r1 and set cpsr's nzcv bits
```

  - Οι εντολές σύγκρισης θέτουν flags μόνο - they do not specify Rd
  - Η μετακίνηση δεδομένων δεν καθορίζει τον Rn

Εκτέλεση υπό συνθήκη

- A useful feature of ARM is that almost all instructions can be conditionally executed
  - ✓ Instruction executed only if appropriate N, Z, C or V bits in the cpsr are set
- Conditions under which the instruction should be executed are postfixed into the instruction using one of the mnemonics (shown on next slide)
  - ✓ If the instruction contains no mnemonic, the instruction is always executed
- • Examples

```
- ADDEQ r0, r1, r2     ; r0 ⇦ r1+r2 only if Z bit in cpsr is set

- ADDEQS r0, r1, r2    ; r0 ⇦ r1+r2 only if Z bit in cpsr is set and update
                       ; the N, Z, C, V bits of cpsr

- ADD r0, r1, r2       ; r0 ⇦ r1+r2 regardless of cpsr bits
- ADDMI r0, r1, r2     ; r0 ⇦ r1+r2 only if N bit in cpsr is set
```

Μνημονικά συνθηκών

Suffix/Mnemonic	Description	Flags tested
EQ	Equal	Z=1
NE	Not equal	Z=0
CS/HS	Unsigned higher or same	C=1
CC/LO	Unsigned lower	C=0
MI	Minus	N=1
PL	Positive or Zero	N=0
VS	Overflow	V=1
VC	No overflow	V=0
HI	Unsigned higher	C=1 & Z=0
LS	Unsigned lower or same	C=0 or Z=1
GE	Greater or equal	N=V
LT	Less than	N!=V
GT	Greater than	Z=0 & N=V
LE	Less than or equal	Z=1 or N=!V
AL	Always	

Εκτέλεση υπό συνθήκη

- Use of conditional execution improves code density *and* performance by reducing the number of forward branch instructions

C code	*without conditional instructions*	*with conditional instructions*
`if (x != 0)`	`CMP r3,#0`	`CMP r3,#0`
`  a=b+c`	`BEQ skip`	`ADDNE r0,r1,r2`
`…`	`ADD r0,r1,r2`	`…`
	`skip`	

- By default, data processing instructions do not affect the condition code flags but the flags can be optionally set by using "S"
  - ✓ CMP does not need "S"

  Loop

  …

  SUBS r1,r1,#1

  BNE loop
- N, Z, C, V bits of cpsr do not get updated by an instruction unless the S bit is set

Παραδείγματα Εκτέλεσης υπό συνθήκη

- Use a sequence of several conditional instructions

```
if (a==0) x=1;
    CMP r0,#0
    MOVEQ r1,#1
```

- Set the flags, then use various condition codes

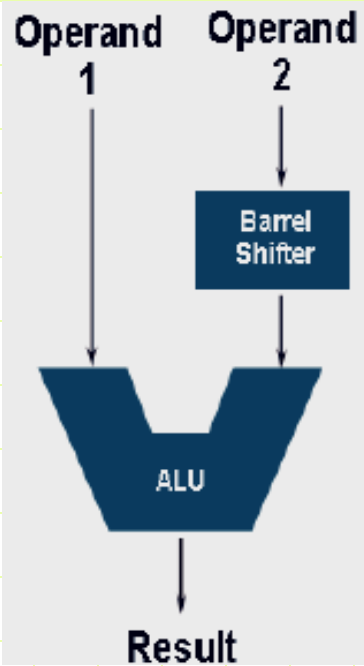
```
if (a==0) x=0;
if (a>0) x=1;
        CMP r0,#0
        MOVEQ r1,#0
        MOVGT r1,#1
```

- Use conditional compare instructions

```
if (a==4 || a==10) x=0;
        CMP r0,#4
        CMPNE r0,#10
        MOVEQ r1,#0
```

Barrel Shifter

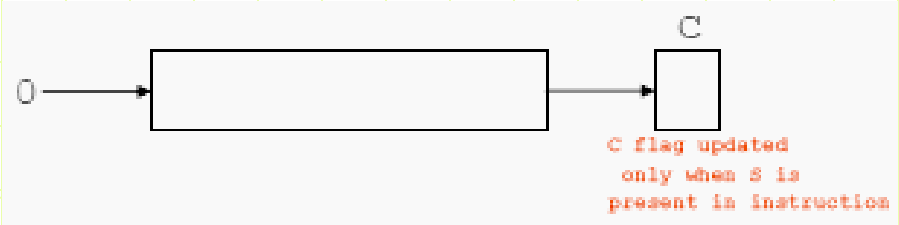
- ARM processors include a barrel shifter that can be used to shift the second operand
- **Shifting can only be applied to the second operand**
- Shift value can be specified by either an immediate constant or as a register
- Instruction do not take any longer to complete
  - ✓ unless the shift is specified by a register
- Operand 2 can be
  - ✓ Register
    - o Shift value can be either be:
      - 5 bit unsigned integer
      - Specified in bottom byte of another register
  - ✓ Immediate constant:
    - o 8 bit constant rotated through an even number of positions



Είδη ολίσθησης

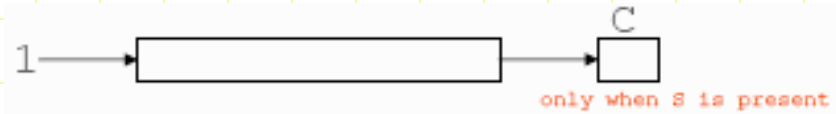
- Logical Shift Left (LSL) by n bits (multiply by 2n)

- Logical Shift Right (LSR) by n bits (unsigned divide by 2n)

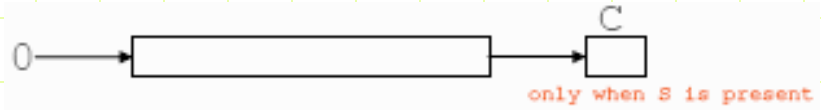


Είδη ολίσθησης– ΙΙ

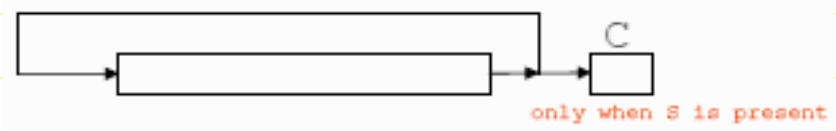
- Arithmetic Shift Right (ASR) by n bits (signed divide by 2n)
  - ✓ If the original most significant bit is 1 (for –ve numbers)



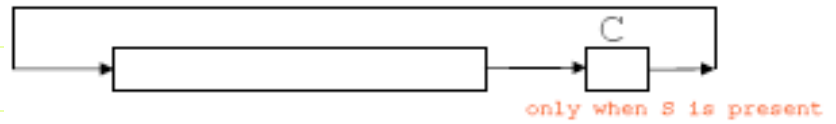
  - ✓ If the original most significant bit is 0 (for +ve numbers)



- Rotate Right (ROR) by n bits



- Rotate Right Extend (RRX)



Χρήση του Barrel Shifter

- Οποιαδήποτε εντολή data processing μπορεί να χρησιμοποιήσει τον barrel shifter να επεξεργαστεί τον δεύτερο τελεστή (shifter operand)
- Shifter operand could be
  - ✓ 1. An 8-bit immediate constant rotated right by 0, 2, 4, 6, …30

    In this case the shifter operand is of the form #constant{,#rotation-amount}

    Examples:
    - MOV r0, #32; r0 ← 32
    - MOV r0, #32, #2; r0 ← 1
    - MOV r1, #0xFF, #4; r1 ← 0xFF000000
  - ✓ A register shifted (LSL, LSR, ASR or ROR) by a constant (from 0 to 31) where the amount of shift is specified by an immediate value

    In this case the shifter operand is of the form register {, shift shift-amount}

    Examples:
    - SUB r0, r1, r2, LSL #2; r0 ← r1-4*r2
    - RSB r0, r1, r2, ASR #2; r0 ← r2/4 – r1

    All multiplications by a constant which is a power of two +/- 1 can be efficiently implemented using the barrel shifter

    Examples:
    - r0 = r1 * 8 MOV r0, r1, LSL #3
    - r0 = r1 * 9
    - = r1+(r1*8) ADD r0, r1, r1, LSL#3

Χρήση του Barrel Shifter

- 3. A register shifted (LSL, LSR, ASR or ROR) by a constant where the amount of shift is specified by another register

  In this case the shifter operand is of the form register {, shift register}

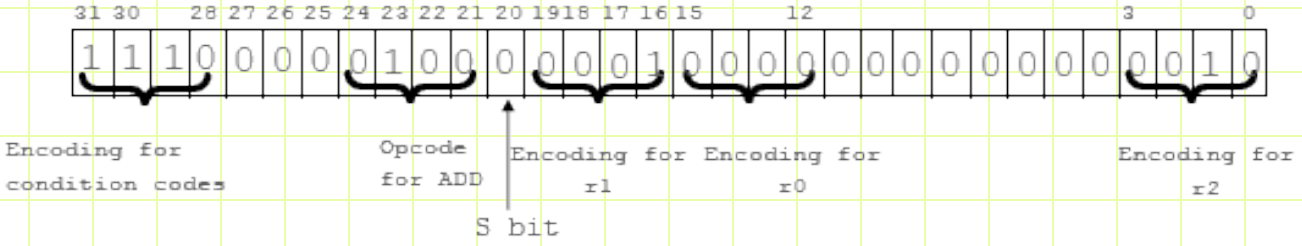
  Examples:
  - ADD r0, r1, r2 ROR r5;
  - SUB r3, r2, r1, LSL r0;
- 4. A register rotate right extended

  In this case the shifter operand is of the form register, RRX

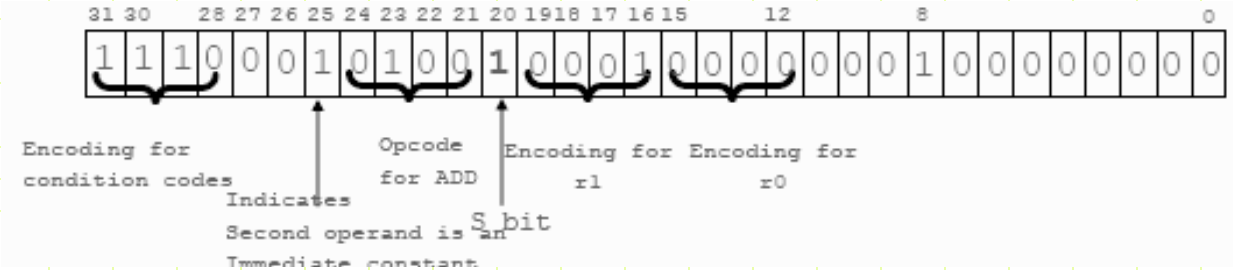
  Examples:
  - MOV r3, r2, RRX;

Κωδικοποίηση εντολών Data Processing

- Binary encoding of instruction
  - ✓ Example
    - ADD r0, r1, r2; r0 ← r1+r2



    - ADS r0, r1, #256; r0 ← r1+r2 and set cpsr's nzcv bits



Σύνοψη

- ARM architecture
  - ✓ Banked registers
  - ✓ Exception handling
  - ✓ Current program status register
- ARM instruction set
  - ✓ Data processing instructions
  - ✓ Conditional Execution of all instructions
  - ✓ Barrel Shifter
- Next lecture
  - ✓ Branch instructions
  - ✓ Load/Store instructions
  - ✓ Program Status instructions