ECE/CS 5780-6780 : Embedded Systems Design
Lect. 03: ARM ISA

Pierre-Emmanuel Gaillardon
Department of Electrical and Computer Engineering – University of Utah

Spring 2017
Salt Lake City, UT, USA

IMFlash TechTalks

TECHTalks

Win $200 and an opportunity to interview with IM Flash. Create a 5 minute presentation on the topic listed at right. Apply by Jan. 25. If your presentation is selected you will present and could win $200

- Feb. 1, 2017 – at IM Flash

"Where do we go from here?
- How does Moore's law apply beyond 20 nm?"

All TECHTalks are at 6:00 PM

To apply and for more info, go to

http://imflash.com/ustar

Submit presentation outlines by Jan 25 to Jake LaMarr, jlamarr@imflash.com

IM Flash, 4000 N Flash Drive, Lehi, UT 84043

FLASH
an intel, micron venture

Objectives for Today!

- **Get familiar with the ARM M processor architecture**
- Understand the architecture of a microcontroller
- **Acquire the fundamentals of the hardware/software interface**
- Acquire the fundamentals for sensing and controlling the physical world
- Learn modeling techniques for embedded system design
- Understand the usage of several peripherals through labs
- Design a complete embedded system – from specs to realization
  - **Realization of a PCB**
  - Behavioral modeling of the system
  - Complete SW realization

ARM Instruction Set Architecture

Instruction Set Architecture

- How to control the processor?
- What is stored in the memory?

Thumb opcodes

A binary-coded instruction that tells the processor which operation to execute

These operations can be directly translated into a Human-readable language called … Assembly

- Different types of assembly instructions
  - Memory access instructions
  - Arithmetic-logic instructions
  - Control (jump) instructions
  - Instructions to manipulate the control register

Memory access instructions

- Two basic operations
  - LDR: Load register content with value from memory address
  - STR: Store register content in memory address
- Syntax:
  - **LDR**{<cond>}{<size>} Rd, <address>
  - **STR**{<cond>}{<size>} Rd, <address>

    Examples: **LDR r0,[r1,#8]**

    STR r0,[r1,#8]

- Memories in the system must support all sizes (default: Word size)
  - <size> = Signed (S) and then Byte (B), Halfword (H)

  Examples:

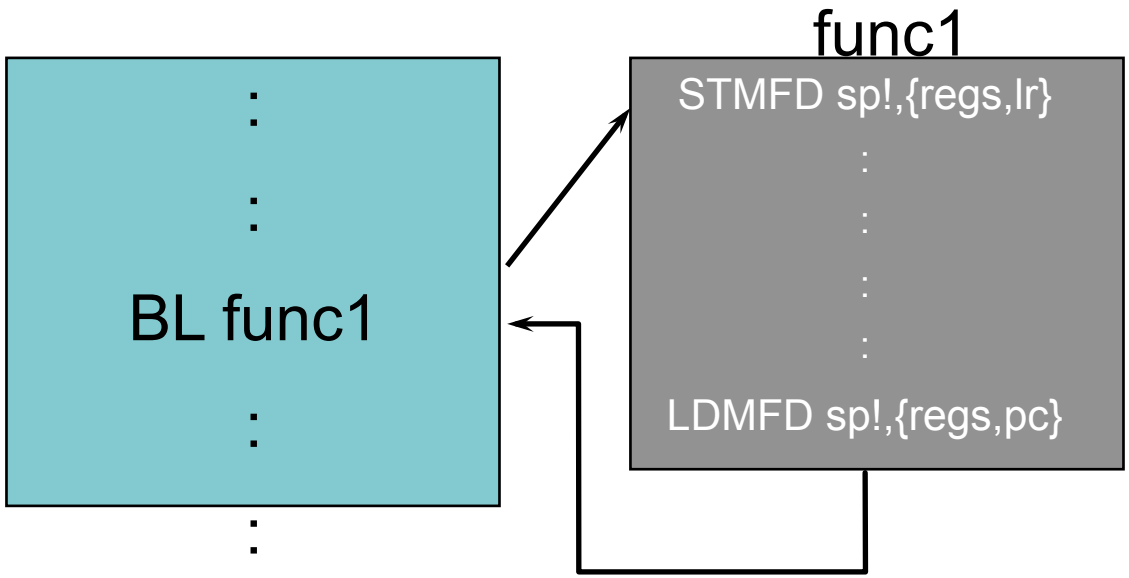
LDR	**STR**	Word
LDRB	**STRB**	Byte
LDRH	**STRH**	Halfword
LDRSB		Load byte with sign
LDRSH		Load halfword with sign

Arithmetic-Logic instructions

- They only work between registers, types:
  - Arithmetic: `ADD` `ADC` `SUB` `MUL` `SBC` `RSB` `RSC`
  - Logic operations: `AND` `ORR` `EOR` `BIC`
  - Comparisons: `CMP` `CMN` `TST` `TEQ`
  - Data movements: `MOV` `MVN`
- Syntax: `<Operation>{<cond>}{S} Rd, Rn, Operand2`
  - Example: **`ADD r0, r1, r2`**

Jumps instructions: branches and subroutines

- **B <label>**
  - Jump that is ***PC*** relative, up to ±32MB range
- **BL <subroutine>**
  - Stores return address in ***LR*** (***r14***)
  - Returning implemented by restoring the ***PC*** from ***LR***



Conditional execution and flags in ARM instructions

- ARM instructions can be made to execute conditionally by post-fixing them using the condition code field: *{<cond>}*
  - This improves code density and performance by reducing the number of forward branch instructions
  - Example: `ADD{<cond>} Rd, Rn, Operand2`

```
  CMP   r3,#0
  BEQ   skip
  ADD   r0,r1,r2
skip:
```



```
CMP   r3,#0
ADDNE r0,r1,r2
```

Accepted conditions in ARM instructions

- ***AL*** is defined by default and it is not needed to indicate it

Suffix	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	

Equivalence between C and ARM assembly

- Definition

```
if(cond) {instTrue;} else {instFalse;}
```

- Uses comparison (*CMP*) and conditional branch (*Bcond*) inst.

```
CMP ifCondition
B<NOT cond> caseFalse
    ; execution instruct. case true
    ....
    B regroup
caseFalse:
    ; execution instructs. case false
    ....
regroup:
    ; insts. with regrouped flow
```

- Example

```
if (a==7) {count+=1;}
else {count-=1;}
…
```

  - *a* is stored in r0 ; *count* is stored in r1

- Solution

```
    CMP r0,#7
    BNE caseFalse
    ADDS r1,r1,#1
    B regroup
caseFalse:
    SUBS r1,r1,#1
regroup:
    …
```

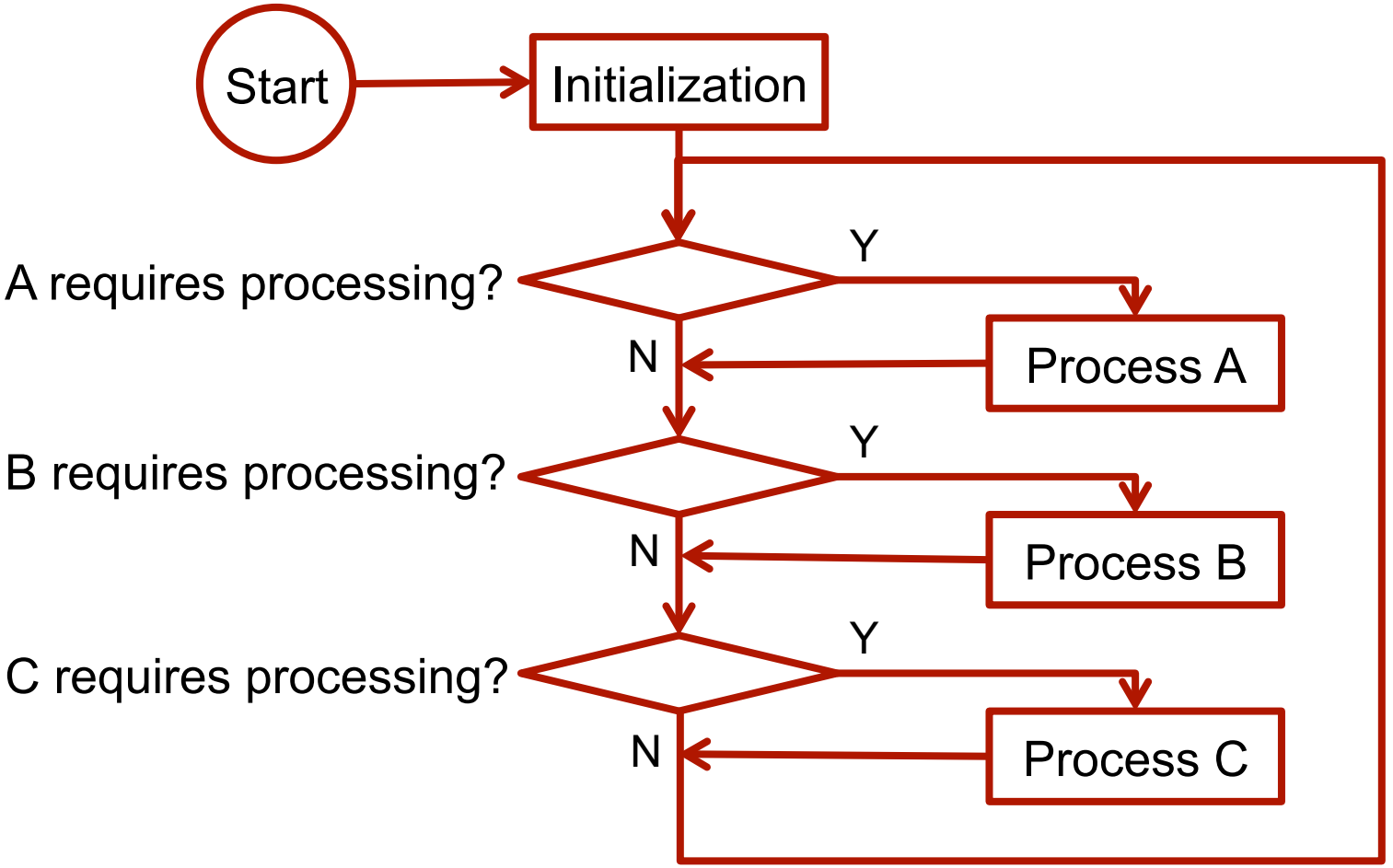
Embedded Systems Software Flows

Embedded Software Program Flows

- Many different ways to structure the flow of the application processing
- Compared to traditional software, the program never ends!
- Four main concepts:
  - Polling
  - Interrupt
  - Hybrid
  - OS

Polling

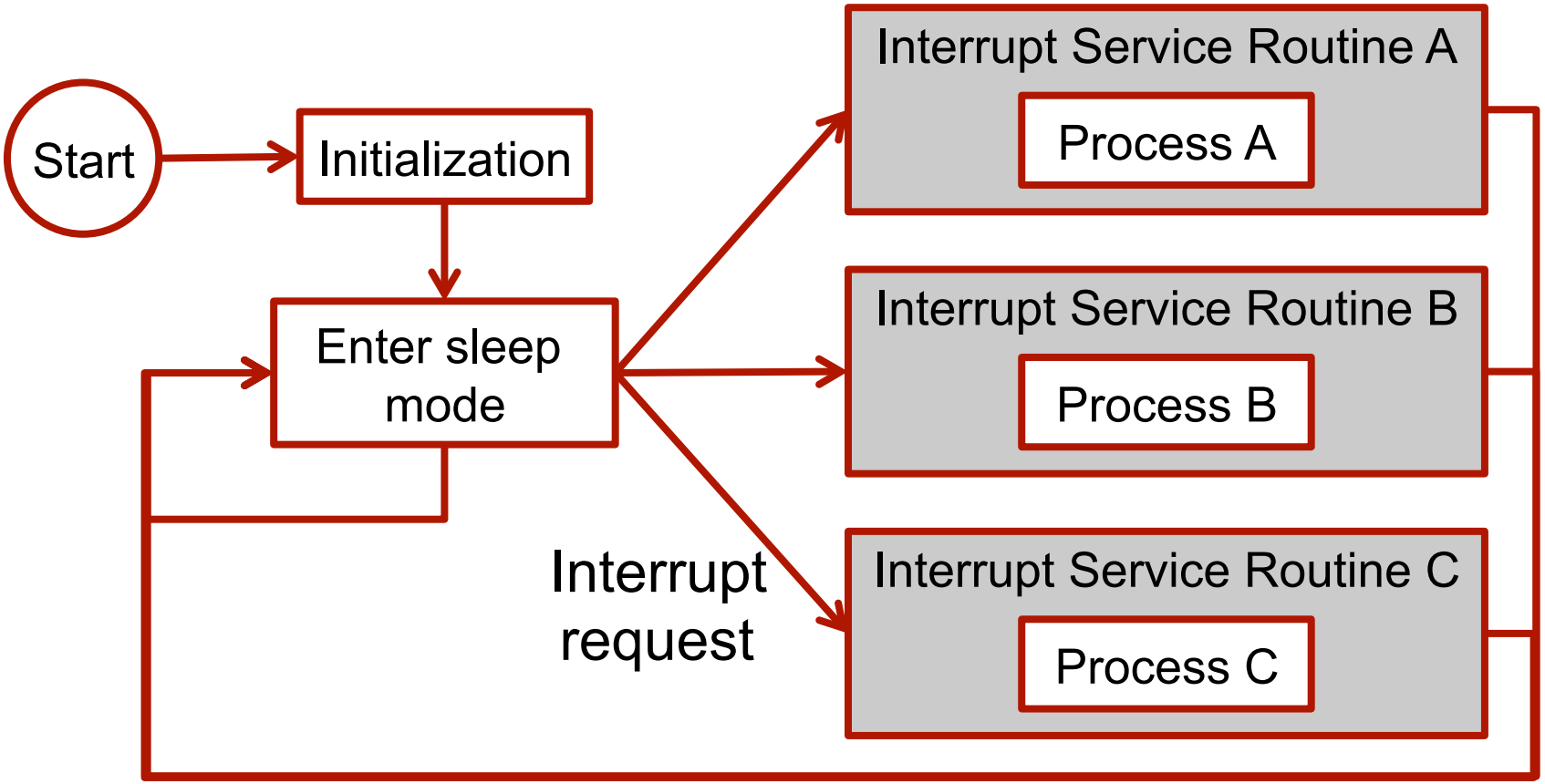
This consists of a giant loop – work good for small applications



If one process takes long time, other process will loose reactivity

Interrupt Driven

Process only on demand – Possibility to grant priorities

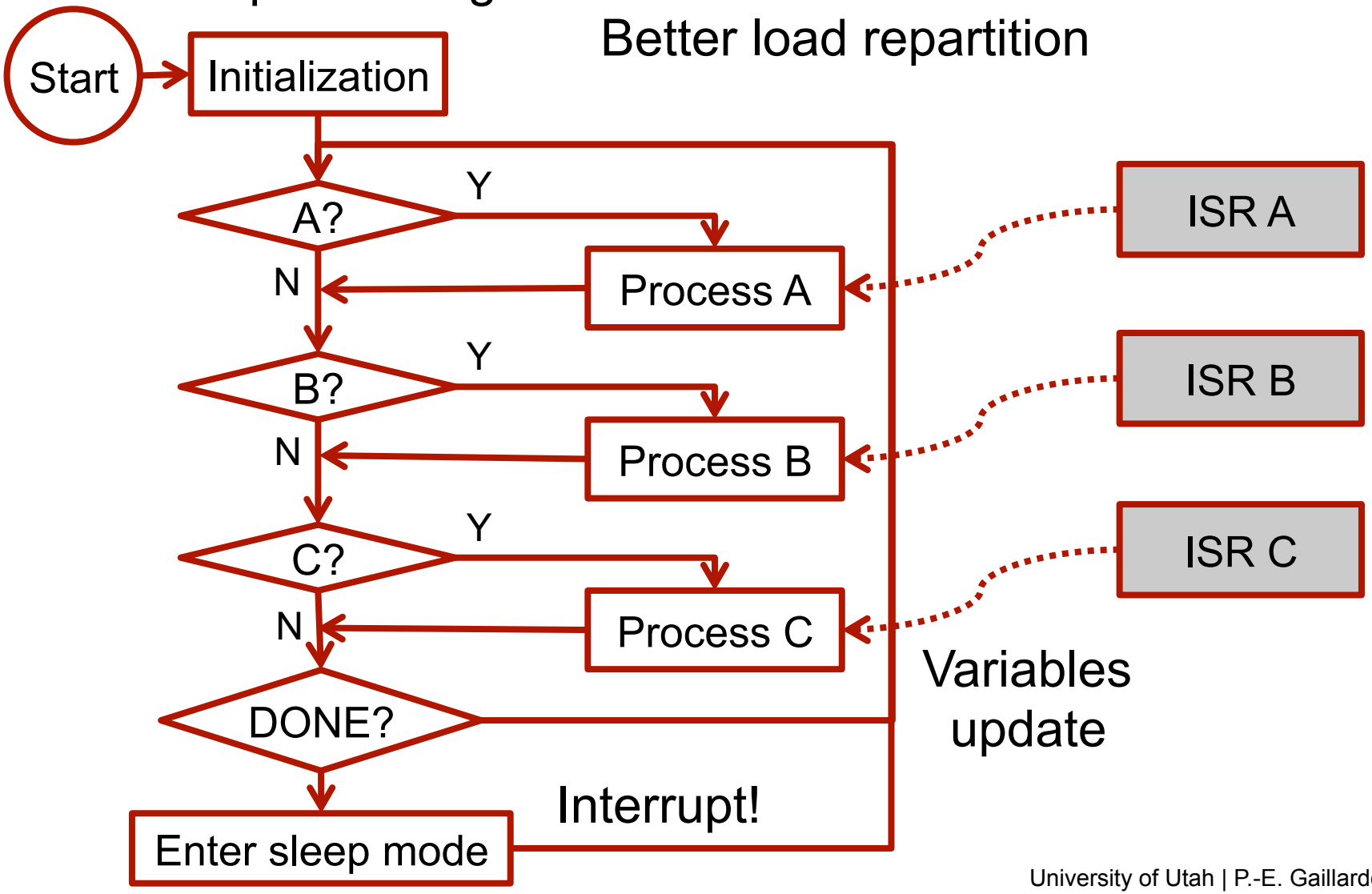


What happens if a high-priority task has a long processing time?

Combination of Polling and Interrupt Driven

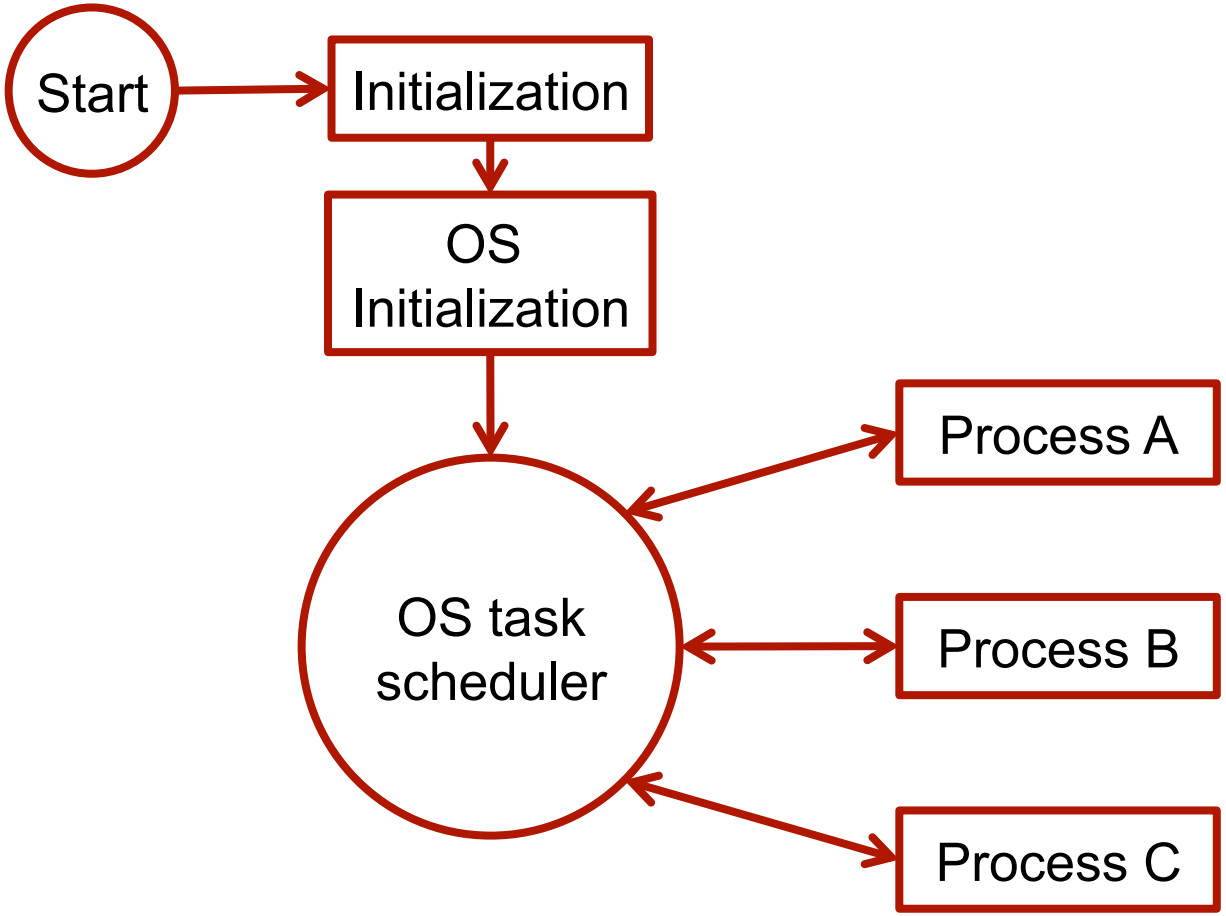
Divide the processing into ISR and main routine

Better load repartition



Handling Concurrent Processes

Use an Operating System



Thank you for your attention

Questions?

Laboratory for NanoIntegrated Systems
Department of Electrical and Computer Engineering
MEB building – University of Utah – Salt Lake City – UT – USA