**Sessió 8: Estructura ATMega328P**

**Gestió PC / Crides a funció: Stack Pointer (SP)**

Diferents modes de funcionament de pila:

1. Apilem de posicions altes de memòria cap a posicions baixes.
2. Apilem de posicions baixes de memòria cap a posiciones altes.

Imagen que contiene Tabla

Descripción generada automáticamente

1. Si apunta a primera posició lliure de la pila
2. Si apunta a darrera posició ocupada de la pila

Escala de tiempo

Descripción generada automáticamente con confianza media

**Funcionament Stack**

Dues operacions:

PUSH 🡪 Posar informació a Stack / moure SP

PULL 🡪 Recuperar informació de Stack / moure SP

1. Apilem cap a munt... SP apunta primera lliure
   1. Push: Guardem dada... pugem SP ADREÇA 🡪 [SP]... SP = SP - 1
   2. Pull: Baixem SP... recuperem dada SP = SP + 1 ... [SP] 🡪 ADREÇA (SP)
2. Apilem cap a munt... SP apunta darrera ocupada
   1. Push: Pugem SP... guardem dada SP = SP – 1 ... ADREÇA 🡪 [SP]
   2. Pull: Recuperem dada... baixem SP [SP] 🡪 ADREÇA (SP) ... SP = SP + 1
3. Apilem cap a baix... SP apunta primera lliure
   1. Push: Guardem dada... baixem SP ADREÇA 🡪 [SP] ... SP = SP + 1
   2. Pull: Pugem SP... recuperem dada SP = SP – 1 ... [SP] 🡪 ADREÇA (PC)
4. Apilem cap a baix... SP apunta darrera ocupada
   1. Push: Baixem SP... guardem dada SP = SP + 1 ... ADREÇA 🡪 [SP]
   2. Pull: Recuperem dada... pugem SP [SP] 🡪 ADREÇA (PC) ... SP = SP – 1

Imagen que contiene Tabla

Descripción generada automáticamente

**Gestió PC / Crides a funció: Stack Pointer (SP).  
Registre 16 bits (SRAM Memòria 1Byte/Posició)**

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Tabla

Descripción generada automáticamente

Stack a SRAM... Memòria 1 byte ample...

Dades... Operands... 1 byte... 1 posició...

Adreces... de memòria de codi... 2 bytes... 2 posicions...

**Gestió PC / Crides a funció: Stack**

Per tant, a Stack guardem les adreces de retorn a les crides a funció **(CALL/RET)**

**De fet podem guardar qualsevol dada que necessitem poder recuperar més tard**

Les **variables locals** de les funcions també les posarem a la pila (les declarem a la pila)

Imagen que contiene Gráfico

Descripción generada automáticamente

A mida que vaig creant variables, les vaig apilant...

Cada variable amb la mida que li correspongui

Les **variables locals** de les funcions també les posarem a la pila

Imagen que contiene Gráfico en cascada

Descripción generada automáticamente

A mida que vaig creant variables, les vaig apilant...

Per tenir la referència de on comencen les variables locals farem servir un punter **(Frame Pointer)**

Imagen que contiene Gráfico

Descripción generada automáticamente

La posició de memòria on està la primera variable estarà apuntada per **FP** que no es mou al llarg de la funció: abans de crear variables **FP = SP.**

Com podem anidar crides, també haurem de guardar FP a Stack abans de fer una crida a una funció.

Diagrama

Descripción generada automáticamente con confianza media

Per tant, a Stack guardem:

1. FP abans de fer crida a funció
2. Adreça de retorn de la crida a la funció (copiem SP 🡪 FP)
3. Variables locals

A mida que guardem informació per recuperar més tard, SP va decrementant el seu valor.

Si tenim moltes crides a funció anidades... si tenim moltes variables locals...

La **pila quedarà desbordada**... (quan passarà això?) 🡪 **STACK OVERFLOW!**

**Blocs de memòria a SRAM**

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente