## Aplicaciones de la Microprogramación

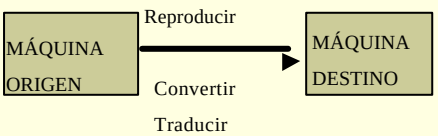
- Realización de Computadores
- Emulación
- Soporte de Sistemas Operativos
- Soporte de lenguajes de alto nivel

## Aplicaciones de la Microprogramación

- Microdiagnóstico
- Adaptación al usuario
- Realización de nuevas instrucciones
- La microprogramación se usa actualmente en periféricos (módems, impresoras, etc)

## ¿Qué es un emulador?

- Emulador = programa software que intenta reproducir el funcionamiento de una computadora (origen) en otra computadora (destino).




## Estructura de un emulador

- Los componentes emulados trabajan en paralelo en el sistema original.
- Algoritmo secuencial.
- Solución: entrelazar la ejecución de los diferentes componentes.

## Emulación de la CPU (1)

- Dos formas de implementarla:
- Emulador intérprete: fetch-decode-execute-loop. Fácil de implementar. Emulación relativamente lenta.
- Solución: threaded code interpreters , implementarlo en ensamblador.

## Emulación de la CPU (2)

- Traducción binaria: convertir el código de la CPU origen a código de la CPU destino.
- Difícil de implementar.
- Emulación muy rápida.
- Dos tipos: estática (traducir binarios) y dinámica (en tiempo de ejecución).

## Emuladores de CPU

- Algoritmo básico: fetch – decode- execute
- Fetch: leer el código de operación (opcode). Tamaño fijo o tamaño variable de opcode.
- Decode: identificar la instrucción y sus operandos. Switch-case. Jump Tables.
- Execute: emulación de la instrucción.

## Instrucción

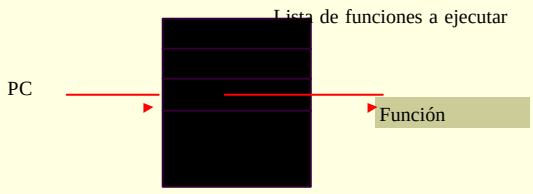
- Esquema básico de la ejecución una instrucción:

```
obtenerOperandos()
realizarOperacion()
guardarResultado()
actualizarCiclosEjecutados()
```

## Threaded code interpreters

- Overhead de un intérprete: cada instrucción se decodifica tantas veces como se ejecuta (costoso).
- Solución: guardar el resultado de la decodificación (dirección de la rutina que emula la instrucción + operandos).
- Threaded Code: Lista/Vector de direcciones de código a ejecutar.

## Threaded code




Se ejecuta código utilizando un índice a una tabla de funciones (instrucciones de la CPU origen decodificadas).

## Ensamblador

- Codificar las instrucciones o todo el emulador en ensamblador para:
  1) Aprovechar mejor las capacidades de la CPU destino (flags).
  2) Implementar más eficientemente el bucle principal (fetch-decode). Se puede poner *inlined* al final de la instrucción.

## Tarjetas Inteligentes tomada de las siguientes autores


**Realizado por:**
Jesús Sánchez Orozco
José R. Sánchez Orozco
Benito Recuero Díaz
Francisco José Seva Mora
Álvaro Rendón Pérez

**Presentado por:**
Francisco J. Seva Mora
Álvaro Rendón Pérez



Marc Sallés Navarro
Juan Antonio Moya Vicén


20/05/2004

## Tarjetas Inteligentes (1)

- Tarjetas inteligentes con microprocesador
  - Tarjetas con un microprocesador capaces de almacenar y procesar información seguramente
  - Son las llamadas tarjetas inteligentes
  - Definición: ISO 7816 / ISO 14443
  - Ventajas:
    - Dispositivo seguro por definición
    - Capaces de procesar información (además de almacenarla)
    - Mayor versatilidad (al poder ser programadas).

## Tarjetas Inteligentes (2)

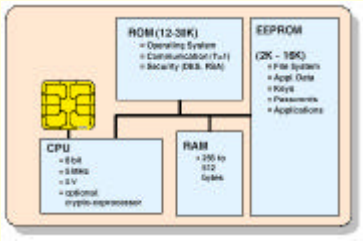
  - "Gran" espacio de memoria (2 KBs, 4 KBs, ...)
  - Pueden ser multiaplicación (monedero electrónico, información bancaria, telefonía, transporte)
- Inconvenientes:
  - Lenguajes de programación de tarjetas dependientes del hardware.
  - Programación de las tarjetas en ensamblador.
  - Aplicaciones desarrolladas exclusivamente por el proveedor de la tarjeta.

## Tarjetas Inteligentes (3)

- Ejemplos de entornos de aplicación:
  - Mundo de la telefonía: tarjetas inteligentes de teléfonos móviles (módulos SIM)
  - Mundo de las comunicaciones: tarjetas inteligentes con claves privadas para acceso seguro
  - Mundo bancario: monedero electrónico (micropagos) y comercio electrónico

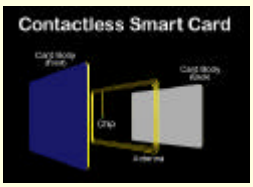
## Tarjetas de Microprocesador (3)

- Arquitectura básica de una tarjeta con microprocesador:




## Tipos de Tarjetas (1)

- Tarjetas inteligentes sin contactos
  - Poseen además del chip una antena de la cual se valen para realizar transacciones
  - Ideales cuando las transacciones tienen que ser realizadas muy rápidamente
  - Definición: ISO 14443




## Tipos de Tarjetas (2)

- Tarjetas inteligentes con contactos
  - Poseen una placa de contactos además del chip
  - Necesitan de un dispositivo lector/grabador para comunicarse con el exterior
  - Definición: ISO 7816
- Tarjetas mixtas
  - Combinación de las anteriores.

## El chip de la Smart Card

**Memory Chip**

- Contenido no modificable una vez creado
- Sin capacidad de cálculo
- No ofrece encriptación

**Chip ASIC**

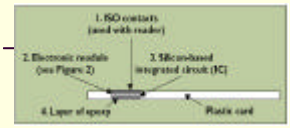
- Contenido no modificable una vez creado
- Limitada capacidad de cálculo
- Ofrece encriptación estática

**Microprocesador**

- Permite modificaciones
- Puede ofrecer diversos servicios para múltiples aplicaciones
- Ofrece encriptación avanzada

## Composición del Chip Microprocesador (I)

- CPU – Interpreta y ejecuta los comandos que el sistema operativo de la tarjeta le facilita.
- ROM – En la ROM se almacena el sistema operativo, así como las funciones de diagnosis y testeo que el fabricante considere necesarias para asegurar el buen funcionamiento de la tarjeta.




## Composición del Chip Microprocesador (II)

- RAM – Se utiliza para almacenar la información temporal. Su contenido se pierde cuando la tarjeta deja de tener alimentación.
- EEPROM – Contiene datos que pueden ser modificados y borrados, como los datos necesarios para el funcionamiento de las aplicaciones de la tarjeta. No es volatil, por lo que sería el equivalente al disco duro en un ordenador convencional. Una Smart Card normal tiene de 8 a 16 kilobytes de memoria de aplicación.

## La interfície

**Contact**

- Requieren contacto físico con la máquina lectora.

**Contactless**

- Integran una antena.
- No requieren contacto con la máquina lectora.
- Las transacciones se realizan mediante campos electro-magnéticos.

## Host y Card Software

**Host software**

- Software diseñado para ser ejecutado en PCs o en servidores de terminales lectores de tarjetas
- Generalmente escrito en un lenguaje de alto nivel (C++, Java, Basic…). Accede a las funcionalidades de la tarjeta mediante librerias y drivers del fabricante

**Card Software**

- Software diseñado para ser ejecutado en la propia tarjeta
- Generalmente escrito en el lenguaje ensamblador del chip de la tarjeta, o en un lenguaje de alto nivel que pueda ser interpretado o compilado para funcionar en la tarjeta.

## Entornos de aplicación reales

- Tarjeta criptográfica CERES (FNMT)

  - Infraestructura de clave pública.
  - 32 Kbytes de EEPROM
  - Almacenamiento seguro de datos sensibles:
    - Claves RSA 1024 bits de firma y confidencialidad.
    - Cifrado simétrico Triple DES.
  - Control de acceso definible para cada fichero.
  - CPU 8 bits / 96 Kbytes de ROM /4 Kbytes de RAM

## Fabricantes Tarjetas Inteligentes


## Fabricantes Tarjetas Inteligentes (3)

## Bibliografía


- Fábrica Nacional de Moneda y Timbre:
  - http://www.cert.fnmt.es/pilotos/tarjeta.htm
  - http://www.fnmt.es/es/html/tace/sc_tage.asp
- M.Mar Group.
  - http://www.exceldata.es/espanol2/espanol/aboutsmart_esp.htm
- ACOTEC (Sistemas Tritón)
  - http://www.acotec.es/triton.htm
- http://www.adventech-mexico.com/index.shtml
- http://www.redox.es
- www.sun.es
- http://www.pcscworkgroup.com/00.html
- http://www.lineasdeltren.com/Lineas12/centro.htm
- http://www.ii.uam.es


## Virus y antivirus

- Un virus es un programa que se oculta (en otros programas) y se copia a sí mismo (se reproduce).
- Muchos de ellos programados en ensamblador.
- Variantes: de gusano, caballos de Troya
- Se suelen “pegar” a archivos COM o EXE, aunque también a algunos de datos: DOC, XLS... Otros se copian en el sector de arranque
- Transmisión por disquetes, redes, correo electrónico

## Redes Neuronales

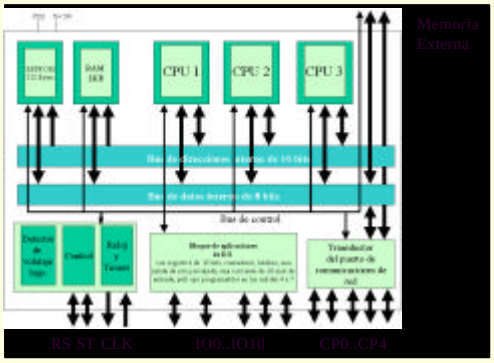
- Esta tecnología desarrollada por *Echelon* es una solución para control distribuido.
- Esto ha incrementado el rendimiento, orden y control de las redes de información.
- Sistemas LonWorks con protocolo LonTalk.
- Un chip neuronal tiene 3 procesadores de 8 bits en *pipeline*.

ECHELON

## LonWorks

- Local Operating Network (LON)
- Utiliza componentes de control y monitoreo inteligentes distribuídos, basado en una red distribuída y tecnología de un sistema orientado a objetos para otorgar soluciones a una amplia gama de aplicaciones de automatización en fábricas y edificios inteligentes. Compartir información de manera transparente.

# Modelo Distribuido




# MODELO DISTRIBUIDO

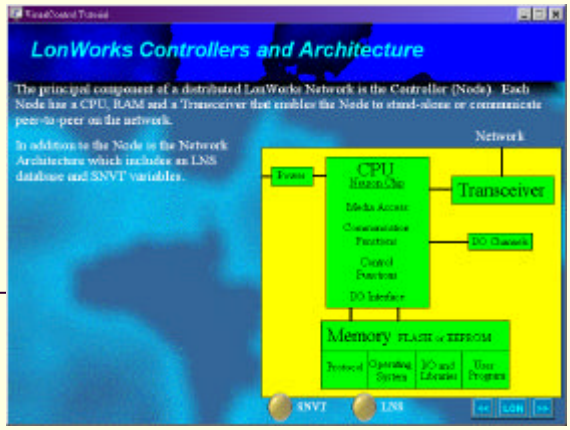
- CPU 1
  - APLICACIÓN
- CPU 2
  - RED
- CPU 3
  - ACCESO AL MEDIO

CARACTERISTICAS

- 8 Bits
- 2 timers de 16 bits
- 11 bits I/O
  - programables con más de 34 aplicaciones
- Posee número de identificación programable de 48 bits

# Modelo OSI en firmware

| OSI | Capa | Descripción | Notas |
|---|---|---|---|
| Física | 1 | Manejo del canal. | - *Medio*: Par trensado Múltiples canales |
| Enlace | 2 | ® Manejo del canal físico.<br>® Comunicaciones confiables y uso adecuado del ancho de banda.<br>® Prioridad | - *Codificación* de datos.<br>- *Verificación* de errores con CRC.<br>- *Evita colisiones*<br>- *Prioridad* opcional y detección de colisiones. |
| Red | 3 | Ruteo, direccionamiento, etc.. | |
| Transporte | 4 | ® Medio de comunicaciones confiables y uso adecuado del ancho de banda<br>® Autentificación | |
| Sesión | 5 | ® Acciones remotas<br>® Autentificación<br>® Administración de red<br>® interfaz de red | |
| Presentación | 6 | ® Interpretación de datos | |
| Aplicación | 7 | ® Compatibilidad de aplicaciones | |




# Características Eléctricas

| | |
|---|---|
| Chip | MC143150FU1BU1 |
| Procesadores | 3 |
| RAM | 2,048 Bytes |
| ROM | NO posee |
| EEPROM | 512 Bytes |
| Contadores / Timers | 2 de 16 bits |
| Interfaz para memoria externa | Si |
| Tipo de empaque | PQFP |
| Pines | 64 |

| Número de pin | Tipo | Función |
|---|---|---|
| A0 – A15 | Salida | Pines de direcciones |
| D0 – D7 | Entrada/Salida | Bus de datos |
| E | Salida | Habilitador (ENABLE) |
| R/W | Salida | Lectura /Escritura (negado) |

# Mapas de memoria

- 512 bytes EEPROM
  - Configuración de red y datos de direccionamiento.
  - Código de identificación único LON 48-bit escribo de fábrica
  - Código de aplicación de usuario

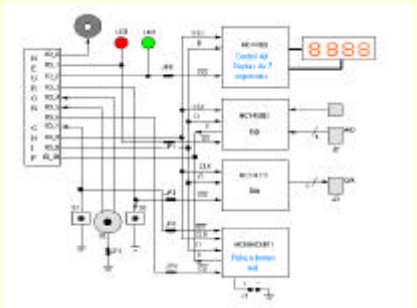
## Mapas de Memoria

- 2,048 bytes RAM
  - Segmento de pila, aplicación y datos del sistema
  - Buffers para el protocolo de red LON y un buffer de aplicación
- 65,536 bytes memoria direccionable total (ROM, EPROM, EEPROM o RAM)

## Mapas de Memoria

- 16, 384 bytes de memoria externa necesarios para sistema operativo de LonWorks
- El resto de la memoria se utiliza para:
  - Código de aplicaciones diseñadas y programadas por el usuario
  - Escritura y lectura de datos adicionales para aplicaciones del usuario.

## Sistema Mínimo Acoplamiento de periféricos



## Aplicaciones de traductores de bajo Nivel en Robots

- Morfología del Robot
- Movimientos del Robot
- Entradas/Salidas
- Programación
- Aplicaciones

## Introducción

**FANUC LTD** se estableció en Julio de 1972 cuando la División de Control Numérico se independizó de FUJITSU.

En 1972 **FANUC** desarrolló su primer robot industrial

**PARA MÁS INFORMACIÓN...**

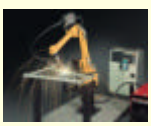
FANUC LTD
Oshino-mura
Yamanashi Prefecture 4010597
Japan

0 DSD0 XQGLDOH' REDO,DFWLYLGDGHV DQXF

**Tel: +81 555 84-5555**
**Fax : +81 555 84-5512**

http://www.fanucrobotics.es

## Tipos de Unidades



**Serie ARC Mate:100iB**
**-Soldadura al ARCO.**

**Serie LR Mate: 100iB**
**-Manipulación.**
**-Carga y descarga de maquina Herramienta.**

**Serie M:410i**
**-Manipulación.**
**-Paletización.**

**Serie A: 521i**
**-Ensamblaje.**
**-Pick-up.**

**6HULH6 : 510i**
**-Soldadura por puntos.**
**-Manipulación.**
**-Todo tipo de aplicaciones.**

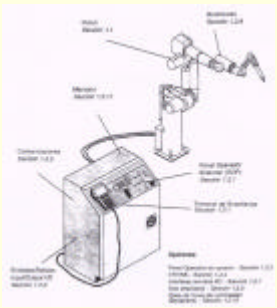
**Serie P: 120**
**-Pintura.**

**6HULHV ( VSHFLDOHV : F200i**
**-Ensamblaje.**
**-Manipulación.**
**-Carga y descarga M.H.**
**-Aplicaciones concretas.**

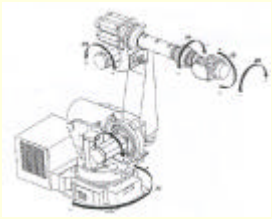
**http://www.fanucrobotics.es**

## Partes del Robot

- El Robot ARC Mate es totalmente articulado con tres ejes principales de rotación y tres ejes secundarios. El elemento terminal es una antorcha fijada en el extremo del brazo del robot, adecuada para la mayoría de aplicaciones de soldadura al arco



## Unidad Mecanica



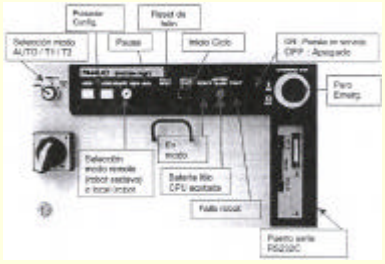
- Ejes principales: eje 1, eje 2, eje 3.
- Ejes secundarios: eje 4, eje 5, eje 6.

## Terminal de Enseñanza

- El terminal de enseñanza es el dispositivo que se utiliza para:
  - -Mover el robot
  - -Crear programas (programas TP)
  - -Comprobar programas
  - -Lanzar en producción
  - -Verificar estados

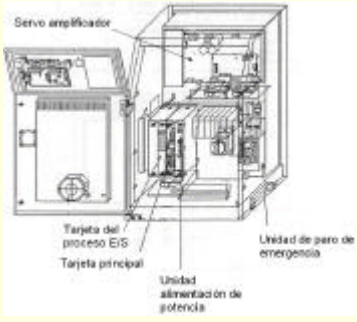
## Panel Operador Estandar

- **E/S Panel Operador.**
  - **–Entradas del Panel Operador de Usuario (UI).**
  - **–Salidas del Panel Operador de Usuario (UO).**



## E/S Armario de Control

- Tarjeta del proceso:
  - Entradas Digitales (DI).
  - Salidas Digitales (DO).
  - Entradas en Grupo (GI).
  - Salidas en Grupo (GO).
  - Entradas Analógicas (AI).
  - Salidas Analógicas (AO).




## Operación de Equipo Remoto en Tiempo Real

**North Carolina State University**

*http://CARL.ce.ncsu.edu/*

- Excavación Computerizada controlada y operada en forma remota, utilizada en situaciones de riesgo
- La calidad del Servicio esta garantizado

## Sistema de Sellado Robotizado

- Realización de un sistema automático de sellado de la sección 18 de la barquilla del A320.
- Aplicar sellante de textura pastosa en las uniones.
- Estación de supervisión donde se define y programa la barquilla.
- Seguimiento de las trayectorias por parte del robot en tiempo real.

- Este proyecto está financiado por la empresa EADS-CASA.

## Programacion

- Lenguaje TPE
  - Basico
  - Trabaja con Registros(Ensamblador)
- Lenguaje Karel
  - Alto nivel

## Introducción

- Con el optimizador de código es posible optimizar los bucles de forma sencilla.
- Considera la estructura segmentada y genera automáticamente código muy paralelizable
- Prestaciones comparables a las alcanzadas con ensamblador clásico.

| | | Eficiencia | Esfuerzo |
|---|---|---|---|
| ASM | Optimiz. Manual | 100% | High |
| Linear ASM | Assembly Optimiser | 95 - 100% | Med |
| C C ++ | Optimising Compiler | 80 - 100% | Low |

## Reglas de Linear Assembly

- Linear assembly es similar al ensamblador clásico pero:
  - No requiere NOPs para completar tiempos muertos.
  - No necesitamos especificar las unidades.
  - La agrupación de instrucciones en paquetes paralelos se realiza automáticamente.
  - Permite el uso de nombres de variables simbólicas.

```
        ZERO    sum
loop    LDH     *p_to_a, a
        LDH     *p_to_b, b
        MPY     a, b, prod
        ADD     sum, prod, sum
        SUB     B0, 1, B0
        B       loop
```

## Generación de Código

- Ficheros .sa

```
_sa_Function    .cproc
                ZERO    sum
loop            LDH     *pm++, m
                LDH     *pn++, n
                MPY     m, n, prod
                ADD     sum, prod, sum
    [count]     SUB     count, 1, count
                B       loop
                .return sum
                .endproc
```

.cproc define elinicio del código

NO usar NOPs
NO usar paralelas
NO especificar unidades func.
NO se requieren registros

.return especifica un valor de retorno

.endproc define el final del código en linear assembly

## Interfasar lenguajes

- Uso conjunto de código C y ensamblador; intrinsics.
- Como norma general: usar código en C para inicialización y código no crítico (en términos de velocidad o tamaño).
- Código crítico en ensamblador o linear assembly.
- Hay tres formas de interfasar C y ensamblador:
  - (1) El código C llama a una función en ensamblador.
  - (2) Una interrupción llama a una función enensamblador.
  - (3) Se puede llamar a una única intrucción de ensamblador utilizando intrinsics.

## Llamar ensamblador desde C

```
main ()
{
      y = asmFunction (a, b);
}

_asmFunction

      b   b3
```

- Funciones C y ensamblador comparten los mismos recursos (registros).
- Además pueden intercambiar datos.
- El código que interfase ambos requiere medidas de compartición de datos e información de control: reglas para el uso de registro compartidos.

## Llamar ensamblador desde C

**main_c.c**

```
int asm_Function (short, short);

short x = 0x4000, y = 0x2000;
int z;

void main (void)
{
     z = asm_Function (x, y);
}
```

- Uso de subrayado " _ " en ensamblador para todas las variables o funciones declaradas en C.
- Las etiquetas tienen que ser globales.

**asm_Function.c**

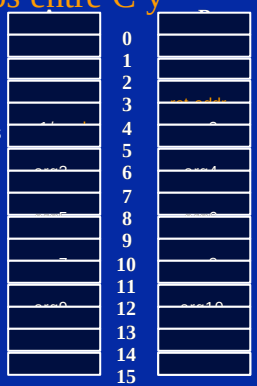
```
int asm_Function (short a, short b)
{
     int y;
     y = (a * b) << 1;
     return y;
}
```

**asm_Function.asm**

```
     .global _asm_Function
```

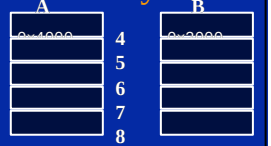
## Paso de Argumentos entre C y Ensamblador

- Los registros siguientes se usan por defecto para pasar y retornar variables cuando se llama una rutina de ensamblador desde C.




## Paso de Argumentos entre C y Ensamblador

- Antes.




- Después.

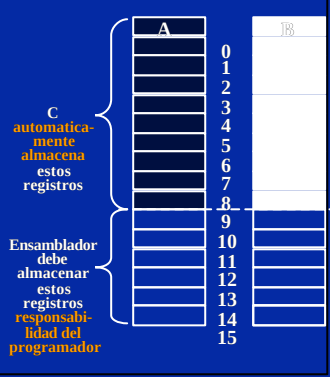

## Paso de Argumentos entre C y Ensamblador

Problemas:

- El código C usará algunos o todos los registros.
- El ensamblador puede requerir el uso de algunos o todos los registros.
- Si no hacemos nada puede que la volver a C algunos de los valores pueden haber sido destruidos.

Solución:

- Códigos C y ensamblador son responsables de almacenar registros si necesitan usarlos.




## Network Processors

# Indice

- Definición
- Evolución
- Tipos
- Diferentes arquitecturas
- Futuro

# Definición

Un *Network Processor* es un ASIP (Aplication-Specific Processor) para el dominio de aplicaciones de red: un dispositivo programable con características de arquitectura y/o trazados de circuitos para procesar paquetes de red.

# Evolución de los NP (1)

- Generación I:
  - Microprocesadores de uso general
  - Basados en software
  - Nuevas características agregadas fácilmente
  - Capacidad de ajuste limitada
  - Fallan en conseguir la velocidad requerida

# Evolución de los NP (2)

- Generación II:
  - ASICs
  - Por hardware (embeded)
  - Muy rápidos
  - Problemas de flexibilidad

# Evolución de los NP (3)

- Generación III:
  - Network Processors
  - Muy flexible (programable)
  - Muy rápido
  - Productos raros
  - No hay standards

# Tipos de NP

- Basados en RISC
  - Muchas instrucciones => más tiempo
  - Intento de paralelismo => incrementa la complejidad del sistema y el tamaño del chip
- Basados en RISC aumentado
  - RISC + ASIC(acelerador hardware)
  - Hereda su inflexibilidad
- Procesador especifico de Red
  - Muchos procesadores pequeños y rápidos.

## Funciones necesarias de los NP

- Capacidad de segmentación y unión
- Reconocimiento de protocolo y clasificación
- Control de colas y accesos
- Control de flujo
- Calidad de servicio (QoS)

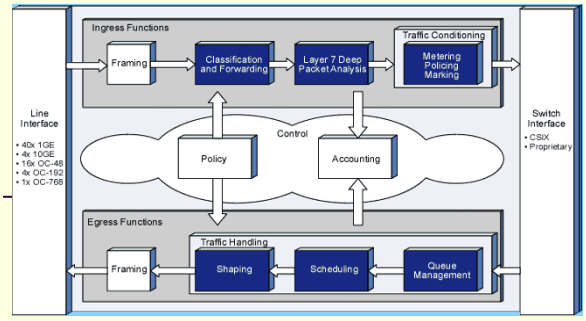
## Arquitecturas

- ClearSpeed
- Intel
- Cisco

## Clear Speed - Características

- Hasta 40 GBits/s
- Sistema de búsqueda en tabla
- Manejo de tráfico programable
- Soporta múltiples protocolos simultáneos (MPLS, IPv4, IPv6...)
- Arquitectura única para todas la funciones (modelo de programación común)
- Viene con un SDK y ADK

## Clear Speed - Arquitectura



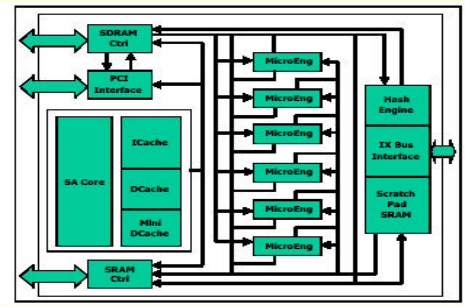

## Clear Speed - Arquitectura

- Arquitectura optimizada
- Con varios procesadores (cada uno con memoria propia) → Procesado paralelo
- Flexible

## Intel - Características

- Diseño realizado por DEC
- 2.5 Mpaquetes/s
- Compuesto por 6 micro-processadoresi un controlador StrongARM
- Processa todos los paquetes de todas las tareas
- Cada processador tiene 4 puntos para hardware suplementario.
- Los 4 puntos de un mismo micro-processador comparten un archivo de registro común

# Intel - Arquitectura



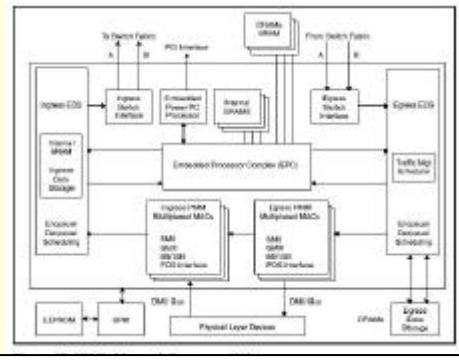

# Intel - Arquitectura

- Los 6 micro-procesadores son programables
- StrongARM coordina las actividades del sistema
- El bus del IX (de 64 bits) conecta micro-procesadores, strongARM, memoria y puede además dar servicio a otros dispositivos como un MAC u otro processador IXP1200

# Cisco - Características

- Suit de Desarrollo con ensamblador, debugger y simulador de sistema
- PowerNP implementado corre a 133MHz, permitiendo Gigabit Ethernet.
- Packet Over SONET

# Cisco - Arquitectura



# Cisco - Arquitectura

- Consiste en un Procesador Embedded Complejo (EPC), hardware de procesado exclusivo de tramas y interfaces para perifericos
- El EPC consta de 7 microprocesadores:
  - almacenamiento de datos, checksum, control de colas, interfaz, copa de cadenas, contador y políticas.