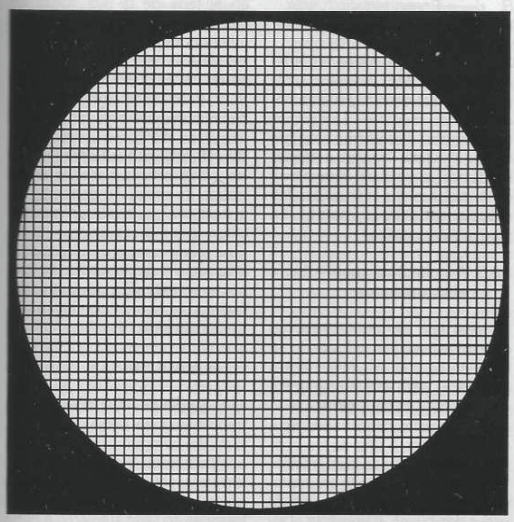
Nuevas arquitecturas de microprocesadores

Máquinas de pilas



❒ **Alvaro Bernal Noreña, M.Sc.**
Ingeniero Electricista
Profesor Asistente
Departamento de Electricidad
Universidad del Valle

RESUMEN

En este trabajo se hace una revisión general de los diferentes tipos de arquitecturas históricamente usadas en el diseño de microprocesadores, con énfasis en las características principales de las arquitecturas de microprocesadores basados en pilas o máquinas de pilas.

Se presentan las ventajas y desventajas, aplicaciones y tipo de implementación de estas máquinas de pilas.
De igual forma, se hace un pequeño análisis de las diferentes formas en que las pilas han sido implementadas en algunos microprocesadores recientemente diseñados con esta arquitectura.

ABSTRACT

In this work a general review of the different microprocessor architectures is presented, more specifically the main features of the stack oriented microprocessors also called stack machines. The advantages, different implementations and more popular applications in the later stack oriented microprocessors are shown.

1. INTRODUCCION

La historia de los microprocesadores es relativamente reciente, pero su evolución ha sido muy rápida. Hablar de esa historia es referirse a las diferentes arquitecturas propuestas para la eficiente implementación de microprocesadores. Muchas de ellas han sido rechazadas u olvidadas, debido a las restricciones tecnológicas de la época. El avance de la tecnología ha permitido el rescate e implementación, con gran éxito, de mu-

chas de esas ideas.
Dependiendo tanto de la aplicación como de la organización funcional de la parte operativa de los microprocesadores, varias arquitecturas han sido concebidas, una de ellas, las arquitecturas basadas en pilas, denominadas comúnmente maquinas de pilas, son el objeto de este estudio.

1.2 Arquitectura general

En general, un microprocesador está compuesto por dos grandes bloques: la parte operativa y la parte de control. La parte de control, muchas veces llamada unidad de control, es la responsable por la activación en el momento apropiado de las señales de control que comandan la parte operativa.
La parte operativa, también conocida como "*data path*" o flujo de datos, es la responsable por el almacenamiento, transferencia y modificación de los datos o direcciones, a través de los elementos de almacenamiento (memorias, registros,...), interconexión (uno, dos, tres buses...), los operadores (unidad lógico aritmética, desplazadores,...) y los elementos de E/S. La organización funcional de la parte operativa del sistema es definida por la forma como estos bloques se interconectan entre sí, por la cantidad de bloques funcionales utilizados y por la estrategia de sincronización del sistema.
Se puede afirmar que la parte operativa de un microprocesador integrado es compuesta por la implementación del hardware necesario para ejecutar el conjunto de instrucciones en el último nivel de interpretación [1].

2. CLASIFICACION DE LA ARQUITECTURA DE LAS PARTES OPERATIVAS

Según el tipo de almacenamiento interno de los datos dentro de la CPU podemos considerar las siguientes opciones:

2.1 Arquitectura con registros de propósito general [1].

Las arquitecturas basadas en registros de propósito general, también llamados de memoria local, se ajustan muy bien para partes operativas que involucran estructuras de dos o tres buses. Estas arquitecturas presentan un formato de micro-instrucción de la forma:

$R_k \leftarrow\!\!- R_i$ op R_j

Donde:
R_k <—— Registro destino.
R_i <—— Primer registro fuente.
R_j <—— Segundo registro fuente.

En este caso sólo se tienen operandos explícitos, independientemente de que sean registros o posiciones de memoria. Las primeras máquinas que usaron este tipo de arquitectura llegaron a utilizar hasta tres campos de direcciones en la palabra de instrucción: las direcciones de dos registros fuente y un registro destino.

2.1.2 Arquitectura que involucra el acumulador

En este tipo de arquitectura, un registro especial es adicionado en la salida de la ULA (unidad lógico aritmética), este esquema permite usar siempre el dato almacenado en este registro denominado acumulador, como a uno de los operandos requeridos en la operación. Esta arquitectura puede ser implementada mediante la utilización de uno o dos buses, y presenta el siguiente formato de instrucción:

$A_c \leftarrow\!\!- A_c$ op R_i

Donde:
A_c ——> Registro acumulador.
R_i ——> Registro fuente.

De esta manera, con uno de los operandos implícitamente localizado en el acumulador, solamente una dirección debe ser especificada en el registro de instrucciones. Con esta arquitectura se tienen instrucciones cortas, minimizando el número de estados internos de la máquina.

2.1.3 Arquitectura basada en pilas

En este tipo de arquitectura los datos son implícitamente almacenados en la parte superior de la pila (tope). En la mayoría de los procesadores que presentan una arquitectura basada en pilas, los operandos son leídos y almacenados en el tope de la pila, todas las operaciones son hechas en los dos primeros registros de la pila, sólo se tiene acceso a la memoria a través de las operaciones PUSH y POP, y todas las otras instrucciones remueven sus operandos de la pila y los substituyen por el resultado, esto implica que a diferencia de los procesadores basados en registros de propósito general y arquitecturas que involucran acumulador, estos procesadores no requieren de campo de direcciones en el registro de instrucciones. Este hecho permite el uso de instrucciones cortas que proporcionan una buena densidad de código [2].

3. MAQUINAS DE PILAS

El concepto de pilas no resulta extraño cuando se habla de arquitectura de computadores. Algunos computadores, y gran cantidad de software, son diseñados con base en ese concepto. Las arquitecturas basadas en pilas surgieron a partir de 1963, cuando Burroughs sacó el sistema B5000, diseñado para ejecutar lenguaje ALGOL [3].
Posteriormente, estudios realizados en el uso de lenguajes de alto nivel, mostraron que las operaciones de llamado y retorno de subrutina eran responsables por el gran consumo de tiempo en la ejecución de programas escritos en lenguajes típicos de alto nivel [4], esto resultaba de la relativa compleji-

dad del hardware utilizado para el control de almacenamiento y transferencia de datos en los registros. El anterior problema era frecuentemente resuelto tanto en las máquinas RISC convencionales como en las arquitecturas orientadas a registros, con el uso de estructuras LIFO, usadas para realizar el almacenamiento de datos, direcciones y variables involucradas en una operación. Este mecanismo hizo que las máquinas de pilas, propuestas a principios de los sesenta, tomaran nuevamente vigencia hasta tal punto que actualmente son ya varios los microprocesadores diseñados con esta arquitectura.

Una pila es una forma de organización y almacenamiento de datos constituida básicamente por un conjunto de registros cuyo contenido, dependiendo del tipo de estructura utilizada, es manipulado mediante desplazamientos de los datos almacenados o mediante transferencia de esos datos entre los diferentes bloques funcionales.

En las arquitecturas basadas en pilas, las pilas son igualmente utilizadas para almacenar datos, los cuales pueden ser cargados directamente, logrando el programador controlar y almacenar explícitamente los datos dentro de ella. Adicionalmente, son también usadas junto con otros bloques funcionales para la ejecución de operaciones aritméticas o de desplazamiento. También pueden conservar mayor espacio de memoria, pues permiten a diferentes subrutinas usar el mismo espacio repetidamente para almacenamiento de variables temporales.

Cuando las pilas dentro de una arquitectura están operando como memorias "*cache*" internas, habilitan el sistema para almacenar un conjunto de códigos de programa, de datos locales y direcciones, esta situación lleva a un descongestionamiento del bus de la memoria principal [5]. Las pilas han sido muy utilizadas en los sistemas para realizar diferentes funciones de computación, la descripción de estas funciones puede ser presentada de la forma siguiente [6]:

- **Evaluación de expresiones**: En este caso las pilas ofrecen manejo automático de resultados intermedios de las expresiones.
- **Almacenamiento de direcciones de retorno de subrutina**: el uso de pilas para el almacenamiento de direcciones de subrutina, resolvió el problema de recursividad en el llamado de subrutinas, así cada vez que una subrutina es llamada, la máquina almacena la dirección de retorno en la pila, esto garantiza que los retornos de subrutina sean procesados en orden contraria a los llamados de subrutinas, así rutinas recursivas pueden llamarse a sí mismas sin problemas.
- **Almacenamiento de variables locales almacenadas dinámicamente**: una pila de variables locales es también utilizada para resolver problemas generados por la recursividad, dada la necesidad de dar al mismo código usos múltiples para diferentes tareas de control, esto significa que cuando se desee hacer el manejo de las variables locales de esta forma, bloques de memoria son localizados en una pila de variables locales, cada vez que una llamada de subrutina es ejecutada.
- **Transferencia de parámetros de subrutinas**: cuando una subrutina es llamada, usualmente la llamada debe estar acompañada con un conjunto de parámetros para ser procesados, los parámetros pueden ser transferidos a registros lo que limitaría su número. Este inconveniente se resuelve copiando los elementos dentro de la pila de parámetros, antes de hacer llamada de subrutinas.

Los métodos desarrollados para el diseño de partes operativas de arquitecturas basadas en pilas, consideran que el dato mas recientemente usado puede ser mas fácilmente accesado e introducen el uso de memorias pequeñas y rápidas dentro del "*chip*" (*on-line*), disminuyendo de esta forma la actividad del bus externo del microprocesador y por tanto resolviendo parcialmente el problema de los grandes atrasos debidos al acceso de la memoria principal [7]. El uso de memoria relativamente pequeñas tipo "*cache*", muy usadas en la arquitectura basada en pilas, resulta en un sistema de memoria con una densidad comparada a las alcanzadas por las memorias dinámicas y con los tiempos de acceso deseados de las memorias estáticas [8]. Mediante el diseño apropiado de las células de memoria, la velocidad relativa y la capacidad de almacenamiento de los diferentes componentes de la jerarquía de memorias pueden ser consideradas para alcanzar un óptimo balance entre desempeño, área de silicio y consumo de potencia.

El mecanismo de detección de situaciones de pila llena "*overflow*" y pila vacía "*underflow*" permite a las pilas un mejor uso del espacio disponible, debido a la implementación de un método basado en el algoritmo conocido como "*Cut-BackK*" [9], este método consiste en transferir de la pila hacia la memoria principal K palabras cuando la pila se encuentre llena y una operación de almacenamiento (PUSH) ocurre. En el caso de la pila encontrarse vacía y presentarse una operación de lectura (POP), las palabras serían leídas de la memoria principal [10]. La inclusión de memorias (pilas) dentro del mismo chip del micro-procesador mejora el desempeño de este, de la velocidad de transferencia de información entre el procesador y la memo-

ria interna depende la velocidad de procesamiento. Idealmente la memoria entonces sería lo suficientemente grande como para almacenar un conjunto de programas y datos suficientes para resolver un problema en particular.

4. ESTIMACION DEL USO DE LAS PILAS

Cuando una arquitectura involucra un sistema de pilas, la primera consideración tiene que ver con su tamaño. Es por esta razón que varios estudios han sido realizados para resolver esta pregunta. La determinación del tamaño de la pila interna en un microprocesador puede ser resuelta con el uso de programas de simulación, donde son considerados varios tamaños.
La simulación realiza mediciones de cantidad de tráfico de información que viene y va a la memoria, a través de las ocurrencias de situaciones de lleno total o vacío total de la pila. En una ocurrencia de "*overflow*" es preciso copiar elementos de la pila a la memoria, y en una operación de "*underflow*" se traerían elementos de la memoria hacia la pila. La forma de atacar el problema de ocurrencias de "*overflow*", puede ser hecha así [6]:

* Tratarlas como fallas del sistema.
* Usar un controlador de demanda de la pila.
* Usar un mecanismo de control de paginación de pilas.
* Usar memorias de datos tipo "*cache*".

El primer caso simplemente asume que una condición de "*overflow*" nunca acontece, implicando el uso de una pila muy grande, ya que el tamaño de la pila debe garantizar que la frecuencia de ocurrencias de "*overflow*" sea mínima, pues de otra forma tratar esta condición como falla reduciría el desempeño del procesador. Esta técnica tiene la ventaja de que el desempeño del sistema es predecible y además no necesita de un hardware de control de pila demasiado complejo.

El segundo caso requiere de un "*stack buffer*", usado circularmente con apuntadores en la posición superior y en la inferior. Además de eso un apuntador que direccione la última posición utilizada en la región de memoria que corresponde a la pila. Cuando acontece un "*overflow*" el elemento alamcenado en la primera posición es transferido a la memoria. En el caso de un "*underflow*" un elemento de la memoria es transferido al "*buffer*". Esta implementación garantiza una mínima cantidad de tráfico en la información ya que el movimiento de los datos es realizado sólo cuando es estrictamente necesario. La mayor desventaja de este esquema es que requiere de un hardware de control complejo, necesitando tres contadores para cada pila.

La tercera opción genera interrupciones cuando existen ocurrencias de "*underflow*" y "*overflow*" y después usa software para controlar las transferencias. Para eso requiere de apuntadores que direccionen el fondo y el tope de la pila. De esta forma cuando el apuntador de la pila es menor que el apuntador del fondo, se origina una transferencia de la memoria de la mitad de los elementos de la pila; en el caso de "*overflow*" ocurre una transferencia de la pila hacia la memoria. Esta técnica usa menos hardware de control requiriendo sólo de registros que almacenan los valores de inicio y tope de la pila, pero requiere un "*stack buffer*" mayor a fin de reducir la frecuencia de las interrupciones.

En el cuarto caso se usa una memoria "*cache*" convencional, lo cual implica una complejidad de hardware importante, pero no presenta las desventajas propias de los métodos descritos anteriormente cuando son usadas en las máquinas basadas en pilas. Usándose este método las necesidades de grandes pilas presentes en los métodos ya descritos, no son necesarios, adicionalmente debido a que la pila "*cache*" está localizada dentro del "*chip*", es posible realizar un acceso mas rápido de los datos recientemente usados, eliminando el consumo de varios ciclos de memoria para realizar acceso a los datos.
Llegando a algunas conclusiones respecto del tamaño de las pilas, se define entonces un "*large stack buffer*" como un sistema de pilas con una profundidad tal que su capacidad no es consumida totalmente en la ejecución de los programas. En general se considera el "*stack buffer*" como grande, cuando cinco o mas niveles de subrutinas pueden ser procesadas sin sobrepasar la capacidad de la pila. En el caso de la pila que es usada sólo para evaluación de expresiones, una profundidad de 16 elementos aproximadamente es considerada grande [11].
Para definir el tamaño apropiado de las pilas, se han usado varias pilas de varios tamaños en la solución de problemas de alta recursividad. En la solución del problema de Hanoi, se concluye que un tamaño de 24 palabras es suficiente para reducir las ocurrencias de "*overflow*" a menos de 1%. Otros autores [12], llegaron a la conclusión que una pila de un tamaño de 32 elementos prácticamente elimina la ocurrencia de "*overflow*" para todos los programas.

5. APLICACIONES

Los procesadores basados en pilas se caracterizan por tener una estructura simple, son muy utilizados para control en tiempo real y aplicaciones en donde es necesaria la rápida conmutación de información. Las áreas mas comunes de aplicación son [13].

Procesamiento de imágenes	Filtros digitales.
Computación gráfica.	Telecomunicaciones.
Electrónica de consumo.	Controladores en robótica.
Control de procesos.	Computadores periféricos.
Control automotriz.	Procesamiento paralelo.

6. CARACTERISTICAS

Las arquitecturas basadas en pilas presentan un modelo simple de evaluación de expresiones, mediante el uso de la "notación reversa". Adicional a la principal característica de este tipo de arquitecturas, que es la de no tener campo de direcciones en su formato de instrucción, tenemos las siguientes:

- Presentan un alto desempeño sin "*pipeline*".
- Tamaño de programa pequeño.
- Rápida ejecución del programa.
- Lógica simple del procesador.
- Baja complejidad del sistema.
- Poco hardware para respuesta de interrupciones.

Las máquinas basadas en pilas ejecutan programas pequeños forzando el uso frecuente de subrutinas para reducir el tamaño de código o para aprovechar las instrucciones cortas de estas máquinas. Con la ejecución de este tipo de programas se obtiene:

- Reducción de costos de memoria.
- Reducción de costos de componentes.
- Reducción de consumo de potencia.
- Mejor velocidad del sistema.
- Mejor desempeño en un ambiente de memoria virtual.
- Requiere menos memoria "*cache*".

Con la reducción de la complejidad del sistema es posible disminuir tanto el tiempo de diseño del sistema, como el tamaño del "*chip*".

7. ESTADO DEL ARTE

Las características de algunos procesadores diseñados con arquitectura basada en pilas, que a continuación se presentan, permiten tener una idea del avance y desarrollo que han tenido estos sistemas en los últimos años [6, 11, 12, 14, 15].

En la tabla 1 se presenta una descripción de cada unidad, la tecnología en que fue implementada, la frecuencia de operación y la forma como fueron concebidas las pilas para la manipulación y almacenamiento de los datos, como también el tamaño y el año en que fueron diseñados. Las expresiones P.D. y P.R. corresponden a la denotación de pila de datos y pila de retorno respectivamente.

8. CONCLUSIONES

Se ha presentado una revisión general de las características y principales ventajas de los microprocesadores con arquitectura basada en pilas. Estas máquinas de pilas han mostrado ser altamente eficientes para la ejecución de lenguajes de programación orientados a pilas, ya que las instrucciones pueden ser implementadas, en la mayoría de los casos, prácticamente en el silicio, osea en su nivel más bajo de interpretación, obteniéndose un excelente desempeño y reducción en la cantidad de hardware comparado con otras arquitecturas.

Tabla 1.

ESTADO DEL ARTE

Unidad	Tecnología	P.D.	PD.	F.	Año
Novix 4016	HCMOS - 3um	Pila dedicada		8 Mhz	1985
WISC CPU/16	TTL - 74LS00	256x16	256x16	280 ns	1986
RTX-2000	CMOS - 2um	256x16	256x16	10 Mhz	1988
MISC M17	HCMOS - 2um	Memoria de Prog		6 Mhz	1988
SF1	CMOS - 3um	128x32	128x32	——	1987
HarrisRTX-32	CMOS - 2.5um	512x32	512x32	8 Mhz	1989
SC32	CMOS - 2um	16x32	16x32	10 Mhz	1989
SWIFT	CMOS - 1.5um	32x32	16x24	42 Mhz	1991
SPOCK	CMOS - 1.2um	32x32	32x32	42 Mip	1993

9. BILIOGRAFIA

[1] ANCEAU, F., The architecture of microprocessors. Addison-Wesley Publishing Company, England, 1986

[2] MILLER, R., Miller J. BREAK-FORTH into FORTH !. Byte Publications, August 1980.

[3] PATTERSON, D.A., HENNESSY J. L. Computer Architecture A Quantitative Approach. Morgan Kauffmann Publisher Inc., 1990.

[4] TAMIR, Y., SEQUIN C., Strategies for Managing the Register File in RISC. IEEE Transaction on Computers, Vol c-32, No. 11, November, 1983.

[5] WEISS, R., RISC Processors: The new wave in ComputersSystems. Computer Design, May 15, 1987.

[6] KOOPMAN, P.J. Jr. Stack Computers - The new wave. Ellis Horwood Limited, John Wiley & Sons, New York, 1989.

[7] KADOTA, H., MIYAKE, J., OKABAYASHI, I., MAEDA, T., OKAMOTO, T., NAKAJIMA, M., KAGAWA, K. A., 32-bits CMOS Microprocessor with On-Chip Cache and TLB. IEEE Journal of Solid State Circuits, Vol sc-22, No 5, October, 1987.

[8] PATTERSON, D.A., SEQUIN, C.H., Design Considerations for Single-Chip Computers of the Future. IEEE Transactions on Computers, Vol C-29, February, 1980.

[9] HASEGAWA, M., SHIGUEI, Y. High Speed Top-of-Stack Scheme for VLSI Processor: a Management Algorithm and its Analysis. Proc. of the 12th Annual International Symposium on Computer Architecture, 1985.

[10] HAYES, R., FRAE-MAN, M.E., WILLIAMS, R.L., ZAREMBA, T., An architecture for the Direct Execution of the FORTH programming Language. Proc. Symposium on Architectural Support for Programming Languages and Operating Systems, ACM, 1987.

[11] LEE, S.C., HAYES, J.R., Development of a FORTH Language Directed Processor using Very Large Scale Integrated Circuitry. John Hopkins APL Technical Digest, Vol 10, Number 3, 1989.

[12] KNOWLES, G., SWIFT: Stack Based Microprocessor for LISP and PROLOG. IEEE Proceeding-E, Vol. 138, No. 5, September, 1991.

[13] DANILE, P., MALINOWSKY C. FORTH Processor Core for Integrated 16-bit Systems. VLSI Systems Design, June, 1987.

[14] WATSON, W., STEPHENS, C., NOVIX - A radical approach to Microprocessor Design. Electronic & Wireless World.

[15] AEDO, E., BERNAL, A., SILVEIRA, R., VAN NOIJE, W., SÁNCHEZ, P.L.P., SPOCK: Um processador experimental de 32 bits. Proc. VII Congresso de Sociedade Brasileira de Microeletrônica, July, 1992.

Todo gran cometido debe tener un principio, pero es en la continuidad hasta el final, hasta que se haya acabado totalmente, donde está la verdadera gloria.

Sir Francis Drake