# MÁ-QUINA Versão 1.1 – 17/04/2012

A máquina definida abaixo é a primeira versão de uma arquitetura com dois objetivos: 1- oferecer simplicidade para uma implementação inicial de simulador de sistema operacional; 2- permitir futuras expansões incluindo a execução de processos de sistema (*shell*, montador, compilador, *kernel*) na própria linguagem da máquina. Para tanto, separam-se as instruções básicas de algumas instruções avançadas (ainda sujeitas a modificações) a serem usadas em trabalhos futuros.

**Processador**

Todas as operações são realizadas em 32 bits. As instruções básicas usam dados e endereços de 8 bits mas definem valores de 32 bits nos registradores, deixando os bytes mais altos com o valor 0 (zero). Os registradores da CPU são:

| | |
|---|---|
| R0 – R14 | 15 registradores de 32 bits para uso geral |
| R15 = SP | ponteiro de pilha de 32 bits |
| PC | endereço da próxima instrução, em 32 bits |
| Z E L | registradores de estado de comparação: zero igual menor (1 bit cada) |
| IR | registrador de instrução de 32 bits |

**Memória:**

Todos os acesso à memória são baseados em palavras, lêem e gravam 32 bits. Os endereços de memória também são baseados em palavras de 32 bits, e não é possível endereçar *bytes* individuais. Na primeira versão a máquina terá somente 1024 palavras de memória.

**Instruções básicas para programas pequenos:**

| | | |
|---|---|---|
| L M r m | r = memory[m] | Load register from Memory |
| L C r c | r = c | Load register from Constant |
| W M r m | memory[m] = r | Write register in Memory |
| S U r1 r2 | r1 = r1 – r2 | SUbtract registers |
| A D r1 r2 | r1 = r1 + r2 | ADd registers |
| D E C r1 | r1 = r1 – 1 | DECrement register |
| I N C r1 | r1 = r1 + 1 | INCrement register |
| C P r1 r2 | if r1 == 0 then Z = 1 else Z = 0<br>if r1 == r2 then E = 1 else E = 0<br>if r1 < r2 then L = 1 else L = 0 | ComPare registers |
| J P A m | PC = m | absolute JumP |
| J P Z m | if Z=1 then PC = m | JumP on Zero |
| J P E m | if E=1 then PC = m | JumP on Equal |
| J P L m | if L=1 then PC = m | JumP on Less |
| I N T n | | software INTerrupt n |

**Entrada e Saída**

As funções de entrada e saída da arquitetura ainda não foram definidas, e serão apenas simuladas pelo código de alto nível do sistema operacional, nesta primeira versão. Operações de leitura e escrita em console ou disco, e termino de processo devem ser solicitadas ao sistema operacional através de uma interrupção de *software* correspondente.

**Código Objeto**

Para facilitar a implementação, a máquina lê *bytes* de instruções que usam códigos correspondente a códigos ASCII de caracteres mnemônicos, e *bytes* de números armazenados em ASCII decimal, os quais devem estar separados por espaços nos arquivos de código objeto. Os espaços simulam espaços reais entre informações no meio físico de armazenamento (fita, disco). O leitor do simulador deve identificar os lexemas começados por algarismos numéricos no arquivo, convertendo-os para *bytes*. Cada palavra de 32 bits no arquivo deve estar separada por uma marca de final de linha, também simulando, por exemplo, separadores em meio físico. Um exemplo de trecho de arquivo objeto seria:

```
J P A 3
0 0 0 2
0 0 0 25
L M 0 1
L M 1 2
A D 0 1
W M 0 1
```

Este programa salta para o endereço 3, carrega o registrador R0 com o valor de memória da posição 1, carrega o registrador R1 com o valor de memória da posição 2, adiciona os registradores 0 e 1, e grava o valor final de R0 na posição de memória 1.

**Instruções avançadas para programas grandes:**

| | | |
|---|---|---|
| L D M r m m m m | Load 32 bit register from memory | |
| L D C r c c c c | Load 32 bit register from constant | |
| W R M r m m m m | Store 32 bit register to memory | |
| L I r1 r2 | r1 = memory[r2] | Load r1 indirectly using r2 |
| W I r1 r2 | memory[r2] = r1 | Write r1 indirectly using r2 |
| J C A 0 m m m m | PC = mmmm | absolute JumP to const |
| J C Z 0 m m m m | if Z=1 then PC = mmmm | JumP to const on Zero |
| J C E 0 m m m m | if E=1 then PC = mmmm | JumP to const on Equal |
| J C L 0 m m m m | if L=1 then PC = mmmm | JumP to const on Less |
| J R A r | PC = r | absolute JumP to reg |
| J P Z r | if Z=1 then PC = r | JumP to reg. on Zero |
| J R E r | if E=1 then PC = r | JumP to reg on Equal |
| J R L r | if L=1 then PC = r | JumP to reg on Less |
| C A L L m m m m | Call 32 bit memory routine | |
| R E T C | Return from 32 bit call | |
| P U S r | Push register into stack | |
| P O P r | Pop register from stack | |

**Instruções para uso em modo protegido do Sistema Operacional**

| | |
|---|---|
| I N 0 0 | Input instructions |
| O U 0 0 | Output instructions |
| M E B r | write MEMory Base register |
| M E L r | write MEMory Limit register |
| R E T I | Return from interrupt |
| I O F F | Interrupt Disable |
| I O N x | Interrupt Enable |
| T A S m | Test And Set memory position m |